EPSON
EXCEED YOUR VISION

TM-S1000 for EMEA

.NET API Reference Guide

Rev.D
M00009103

Cautions

- ❏ No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Seiko Epson Corporation.
- ❏ The contents of this document are subject to change without notice. Please contact us for the latest information.
- ❏ While every precaution has taken in the preparation of this document, Seiko Epson Corporation assumes no responsibility for errors or omissions.
- ❏ Neither is any liability assumed for damages resulting from the use of the information contained herein.
- ❏ Neither Seiko Epson Corporation nor its affiliates shall be liable to the purchaser of this product or third parties for damages, losses, costs, or expenses incurred by the purchaser or third parties as a result of: accident, misuse, or abuse of this product or unauthorized modifications, repairs, or alterations to this product, or (excluding the U.S.) failure to strictly comply with Seiko Epson Corporation's operating and maintenance instructions.
- ❏ Seiko Epson Corporation shall not be liable against any damages or problems arising from the use of any options or any consumable products other than those designated as Original EPSON Products or EPSON Approved Products by Seiko Epson Corporation.

Trademarks

EPSON® is registered trademarks of Seiko Epson Corporation.

EPSON TM-S1000 Driver is based in part on the work of the Independent JPEG Group.

libtiff

> Copyright (c) 1988-1997 Sam Leffler
> Copyright (c) 1991-1997 Silicon Graphics, Inc.
>
> Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.
>
> THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
>
> IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Celeron®and Pentium® are registered trademarks of Intel Corporation.

MS-DOS®, Microsoft ®, Win32®, Windows®, Windows NT®, Windows Vista®, Windows Server®, Visual Studio®, Visual Basic®, Visual C++ ® and Visual C# ® are trademarks or registered trademarks of Microsoft Corporation in the US or other countries.

InstallShield® is trademarks or registered trademarks of Macrovision Corporation in the US or other countries.

PC/AT® is trademarks of International Business Machines Corporation in the United States.

General Notice: Other product and company names used herein are for identification purposes only and may be trademarks of their respective companies.

Contents

Chapter 1
TM-S1000 .NET API Overview

Introduction

This guide provides information necessary for application development using the TM-S1000 .NET API. Refer to "TM-S1000 API Reference Guide" for the following information; functions of the TM-S1000, features of the TM-S1000 API, Installation and Uninstallation, programming method, sample program explanations, and log collection function.

Contents

- ❏ Chapter 1 — TM-S1000 .NET Driver Overview
  Describes how this guide is constructed, and explains about operating environment and the sample programs.
- ❏ Chapter 2 — Creation of the Development Environment
  Explains how to configure VISUAL STUDIO.
- ❏ Chapter 3 — Using .NET API
  Explains how to use the TM-S1000 .NET API, and describes about provided CLASS and constants.
- ❏ Chapter 4 — Reference
  Describes API reference. Also explains about error handling and device information.

Operating Environment

OS

- ❏ Microsoft Windows 8 (32/64 bit)
- ❏ Microsoft Windows 7 SP1 (32/64 bit)
- ❏ Microsoft Windows Vista SP2 (32/64 bit)
- ❏ Microsoft Windows XP SP3 (32 bit)
- ❏ Microsoft Windows 2000 SP4
- ❏ Microsoft Windows Server 2012
- ❏ Microsoft Windows Server 2008 R2 SP1
- ❏ Microsoft Windows Server 2008 SP2 (32/64 bit)

.NET Framework

- ❏ .NET Framework 1.1
- ❏ .NET Framework 2.0
- ❏ .NET Framework 3.0
- ❏ .NET Framework 3.5
- ❏ .NET Framework 4.0
- ❏ .NET Framework 4.5

Computer

30 dpm model/ 60 dpm model

CPU: Pentium 4 1.2 GHz or more

Memory: 256 MB or more than the minimum system requirements of the OS

60 dpm model (when using IQA or Barcode)/ 90 dpm model

CPU: Pentium 4 2.0 GHz or more

Memory: 512 MB or more

Interface

USB2.0

Development Language

Development Tool	Development Language
Visual Studio .NET 2003	Visual Basic .NET 2003
	Visual C++ .NET 2003
	Visual C# .NET 2003
Visual Studio 2005	Visual Basic 2005
	Visual C++ 2005
	Visual C# 2005
Visual Studio 2008	Visual Basic 2008
	Visual C# 2008
Visual Studio 2010	Visual Basic 2010
	Visual C# 2010

Sample Programs

TM-S1000 API includes sample programs that are provided for each development language. Sample programs written in Visual C# and Visual Basic .NET are provided under the .NET environment.

There are eight sample programs. For explanations about the sample programs, refer to Chapter 3 of TM-S1000 API Reference Guide. The explanations are made for C++ sample programs, however, the same procedures apply to the sample programs written in the other languages.

Step 1: Opening/Closing the device
Describes how to execute the device opening/closing process and reading process.

Step 2: Displaying read data
In addition to Step 1, describes how to display read image and MICR data on the application screen.

Step 3: Continuous reading/Electric endorsement
In addition to Step 2, describes how to set the processing methods (continuous reading/one-by-one reading) and electric endorsement.

Step 4: Process setting when a reading error occurs
In addition to Step 3, describes process setting such as how to sort documents into the two pockets automatically when a reading error occurs or how to process differently depending on a read result in an application.

Step 5: Selecting the MICR font and setting the image quality
In addition to Step 4, describes selecting a MICR font and setting the image quality.

Step 6: Reading OCR-A/B font and buzzer setting
In addition to Step 5, describes how to read an OCR-A/B font with the OCR function and buzzer setting.

Step 7: Confirming the Device status and error handling
In addition to Step 6, describes how to confirm the device status, how to handle errors (pocket near-full/paper jam error), and how to process MICR cleaning.

Step 8: Decoding a barcode, confirming the IQA and Waterfall process
In addition to Step 7, describes how to decode a barcode, how to confirm the IQA and how to process Waterfall.

Chapter 2
Creation of the Development Environment

This chapter explains how to create the environment such as VISUAL STUDIO for developing the application. The sample programs include the development environment.

Configuring references

Before developing the application, configure references in VISUAL STUDIO.

Note

When moving the TM-J9000 application, its reference configuration must be replaced with DLL provided for the TM-S1000 API.The following explains the reference configuration procedure using C# of VISUAL STUDIO 2005. The same procedure is required when using VISUAL STUDIO 2003 or VB.NET.

Configure references using the following procedures.

1. Launch a new project.
2. Right click on [Reference Configuration] in Project Explorer, and select [Add reference].

   Note

   When moving the TM-J9000 application, delete the current DLL settings before adding new references.

3. The Add Reference screen is displayed. Click on the Reference tab and select "EpsonBankDriver.dll", then click [OK]. The "EpsonBankDriver.dll" has been stored in / Sample programs/Assembly.
4. Open the Solution Explorer to be sure that the component file "EpsonBankDriver.dll" added at step 3 has been properly added in the project.

This is the end of the reference configuration procedure for .NET environment.

Declaration Statement

The component file configured as reference must be declared to be used for actual programming. The declaration statement differs between C# and VB.NET.

When using C#

The following declaration should be added in declaration section of the source code.

Declaration: using com.epson.bank.driver;

When using VB.NET

The following declaration should be added in declaration section of the source code.

Declaration: Imports com.epson.bank.driver

Chapter 3
Using .NET API

This chapter explains how to use the TM-S1000 API under the .NET environment, and about provided CLASS and constants. For information on how to develop the application using the TM-S1000 API, refer to TM-S1000 API Reference Guide - Chapter 3 “Programming guide” and sample programs.

How to use API

Method and Property

The TM-S1000 .NET API consists of Method and Property. Some of the functions provided in the Method and Property are similar to each other, however, the way to use the functions are different. For more details on the API, refer to Chapter 4 "Reference".

How to use Method

Specify parameters for the Method to be used. See "Constant number list" on page 3-10 for information on the parameters provided for each API. API execution results are set in the “*ErrorCode*”. See Chapter 4 "Reference" for information on the API execution results.

ErrorCode = Method (*Parameter1, Parameter2, ...*);

How to use Property

Use the Property as described below.

1. Acquire or set a value using the Property. See "Constant number list" on page 3-10 for information on the values.

   Acquire a value: Value = <CLASS>.Property;

   Set a value: <CLASS>.Property = Value;

2. API execution results are set in the LastError property. See "LastError" on page 4-96 for information on the API execution results.

   ErrorCode = <MFDevice CLASS>.LastError;

Example of use

Acquiring device status

Only Property is provided for this function.

Property

1. state is used to check the status information.
   ASB *state* = <MFDevice>.Status;
2. result is used to check the execution result.
   ErrorCode *result* = <MFDevice>.LastError;

Acquiring offline cause

Only Property is provided for this function.

Property

1. offline is used to check the offline cause.
   Byte[] *offline* = <MFDevice>.OfflineCode;
2. result is used to check the execution result.
   ErrorCode *result* = <MFDevice>.LastError;

Setting transaction ID

Both Method and Property are provided for this function.

Method

Using SetTransactionNumber of MFDevice CLASS, set the transaction ID to 10. The "*result*" will show the execution result.

ErrorCode *result* = <MFDevice>.SetTransactionNumber(10);

Property

1. TransactionNumber is set to 10.
   <MFDevice>.TransactionNumber = 10;

2. result is used to check the execution result.
   ErrorCode *result* = <MFDevice>.LastError;

Checking transaction ID

Both Method and Property are provided for this function.

Method

1. Reset the variable that the transaction ID is set.

   int *number* = 0;

2. Using GetTransactionNumber of MFDevice CLASS, check the set transaction ID. The "*number*" in the first parameter will show the transaction ID number. And the "*result*" will show the execution result.

   ErrorCode *result* = <MFDevice>.GetTransactionNumber(out *number*);

Property

1. number is used to check the TransactionNumber.
   int number = <MFDevice>.TransactionNumber;

2. result is used to check the execution result.
   ErrorCode *result* = <MFDevice>.LastError;

Setting start positions for attaching electronic endorsement

Both Method and Property are provided for this function.

Method

Using SetPrintPosition of MFDevice CLASS, specify the start position for attaching electronic endorsement in horizontal direction to the first parameter, and specify that in vertical direction to the second parameter.
The "result" will show the execution result.

ErrorCode *result* = <MFDevice>.SetPrintPosition(10, 50);

Property

1. Prepare variables in accordance with the format provided by .NET API, then set the start position for attaching electronic endorsement in horizontal direction to the first parameter, and set that in vertical direction to the second parameter.

   Point *printPosition* = new Point(10, 50);

2. Set 10 for the horizontal start position, and set 50 for the vertical start position.

   <MFDevice>.PrintPosition = *printPosition*;

3. result is used to check the execution result.
   ErrorCode *result* = <MFDevice>.LastError;

Checking start positions for attaching electronic endorsement

Both Method and Property are provided for this function.

Method

1. Reset the variables to which the start positions for attaching electronic endorsement are set.

   int *positionX* = 0;
   int *positionY* = 0;

2. Using GetPrintPosition of MFDevice CLASS, acquire the start positions for attaching electronic endorsement.The first parameter "positionX" will show the horizontal start position and the second parameter "positionY" will show the vertical start position. And the "result" will show the execution result.

   ErrorCode *result* = <MFDevice>.GetPrintPosition(out *positionX*, out *positionY*);

Property

1. Prepare variables in accordance with the format provided by .NET API, acquire the start positions for attaching electronic endorsement.

   Point *printPosition* = <MFDevice>PrintPosition;

2. result is used to check the execution result.
   ErrorCode *result* = <MFDevice>.LastError;
3. The "printPosition.X" will show the horizontal start position and the "printPosition.Y" will show the vertical start position.

   int *positionX* = *printPosition*.X;
   int *positionX* = *printPosition*.Y;

Event Callback Registration Method

Under .NET environment, events are provided instead of the CALLBACK functions provided under Win32 environment.
For more details on the events, refer to "Events" on page 4-101. For more details on the CALLBACK functions, refer to TM-S1000 API Reference Guide - Chapter 3 “Programming guide”.

Device status

Definition

ErrorCode SetStatusBack()

Delegate Definition

- delegate void StatusCallbackHandler(ASB status)
- delegate void StatusCallbackHandlerEx(ASB status, String portName)

Event Definition

- event StatusCallbackHandler StatusCallback
- event StatusCallbackHandlerEx StatusCallbackEx

Example of use

Application callback function definition

```
public void AppStatusHandler(ASB status)
{

}
```

Registration procedure

1. Registration procedure

   ```
   <MFDevice>.StatusCallback +=
                              new MFDevice.StatusCallbackHandler( AppStatusHandler );
   ```

   This does not enable an event to be issued.

2. Function that enables an event to be issued

   ```
   <MFDevice>.SetStatusBack();
   ```

   Calling this function allows an event to be issued.

Cancel procedure

1. Disables an event to be issued.

   ```
   <MFDevice>.CancelStatusBack();
   ```

2. Cancels the registered callback function.

   ```
   <MFDevice>.StatusCallback -=
                              new MFDevice.StatusCallbackHandler( AppStatusHandler );
   ```

Processing status

Definition

ErrorCode SCNMICRSetStatusBack()

Delegate Definition

```
delegate void SCNMICRStatusCallbackHandler( int transactionNumber,
                                            MainStatus mainStatus,
                                            ErrorCode subStatus,
                                            String portName )
```

Event Definition

event SCNMICRStatusCallbackHandler SCNMICRStatusCallback

Example of use

Application callback function definition

```
public void AppSCNMICRStatusHandler( int transactionNumber,
                                     MainStatus mainStatus,
                                     ErrorCode subStatus,
                                     String portName )
                                     {

                                     }
```

Registration procedure

1. Registration procedure

   ```
   <MFDevice>.SCNMICRStatusCallback +=
       new MFDevice.SCNMICRStatusCallbackHandler( AppSCNMICRStatusHandler );
   ```

   This does not enable an event to be issued.

2. Function that enables an event to be issued

   ```
   <MFDevice>.SCNMICRSetStatusBack();
   ```

   Calling this function allows an event to be issued.

Cancel procedure

1. Disables an event to be issued.

   ```
   <MFDevice>.SCNMICRCancelStatusBack();
   ```

2. Cancels the registered callback function.

   ```
   <MFDevice>.SCNMICRStatusCallback -=
       new MFDevice.SCNMICRStatusCallbackHandler( AppSCNMICRStatusHandler );
   ```

API Comparison Chart

For information on how to develop application for using TM-S1000 under .NET environment, refer to the following tables and TM-S1000 API Reference Guide - Chapter 3 “Programming guide”.

Win32 API	.NET API
int BiOpenMonPrinter(int, LPSTR);	ErrorCode OpenMonPrinter(OpenType, String)
int BiSetMonInterval(int, WORD, WORD);	ErrorCode SetMonInterval(Int32, Int32)
int BiGetStatus(int, LPDWORD);	Status
int BiSetStatusBackWnd(int, long, LPDWORD);	ErrorCode SetStatusBack()
int BiSetStatusBackFunction(int, int (CALLBACK EXPORT *pStatusCB) (DWORD dwStatus));	StatusCallback
int BiSetStatusBackFunctionEx(int, int (CALLBACK EXPORT *pStatusCB) (DWORD dwStatus, LPSTR lpcPortName));	StatusCallbackEx
int BiCancelStatusBack(int);	ErrorCode CancelStatusBack()
int BiResetPrinter(int);	ErrorCode ResetPrinter()
int BiGetCounter(int, WORD, LPDWORD);	ErrorCode GetCounter(CounterIndex, Boolean, out Int32&)
	ErrorCode GetCounter(Byte, out Int32&)
int BiResetCounter(int, WORD);	ErrorCode ResetCounter(CounterIndex)
	ErrorCode ResetCounter(Byte)
int BiCancelError(int);	ErrorCode CancelError()
int BiGetType(int, LPBYTE, LPBYTE, LPBYTE, LPBYTE);	ErrorCode GetType (out Byte&, out Byte&, out Byte&, out Byte&)
int BiGetOfflineCode(int, LPBYTE);	Byte() OfflineCode
int BiGetOfflineCodeByIndex(int, int, LPBYTE);	ErrorCode GetOfflineCodeByIndex(Int32, out Byte&)
int BiMICRSelectDataHandling(int, BYTE, BYTE, BYTE);	ErrorCode MICRSelectDataHandling (CharSelect, DetailSelect, ErrorSelect)
int BiMICRGetStatus(int, LPBYTE);	ErrorCode MICRGetStatus(out Byte&)
int BiMICRCleaning(int);	ErrorCode MICRCleaning()
int BiSCNSetImageQuality(int, BYTE, char, BYTE, BYTE);	ErrorCode SCNSetImageQuality (ColorDepth, Double, Color, ExOption)
int BiSCNSetImageFormat(int, BYTE);	ErrorCode SCNSetImageFormat(Format)
int BiSCNSetScanArea(int, BYTE, BYTE, BYTE, BYTE);	ErrorCode SCNSetScanArea(Rectangle)
	ErrorCode SCNSetScanArea(Int32, Int32, Int32, Int32)
int BiSCNGetImageQuality(int, LPBYTE, char *, LPBYTE, LPBYTE);	ErrorCode SCNGetImageQuality (out ColorDepth&, out Double&, out Color&, out ExOption&)
int BiSCNGetScanArea(int, LPBYTE, LPBYTE, LPBYTE, LPBYTE);	ErrorCode SCNGetScanArea(out Rectangle&)
	ErrorCode SCNGetScanArea(out Int32&, out Int32&, out Int32&, out Int32&)

Win32 API	**.NET API**
int BiSCNSetCroppingArea (int, BYTE, BYTE, BYTE, BYTE, BYTE);	ErrorCode SCNSetCroppingArea (Byte, Int32, Int32, Int32, Int32)
	ErrorCode SCNSetCroppingArea(Byte, Rectangle)
int BiSCNGetCroppingArea(int, LPWORD, LPBYTE);	ErrorCode SCNGetCroppingAreas(out Byte()&)
	ErrorCode SCNGetCroppingAreas(out Hashtable&)
int BiSCNDeleteCroppingArea(int, BYTE);	ErrorCode SCNDeleteCroppingArea(Byte)
int BiSCNSelectScanUnit(int, BYTE);	ErrorCode SCNSelectScanUnit(ScanUnit)
int BiSCNMICRFunction (int iHandle, LPVOID lpvStruct, WORD wFunction);	ErrorCode SCNMICRFunction(FunctionType)
	ErrorCode SCNMICRFunction(MF, FunctionType)
int BiSCNMICRCancelFunction (int iHandle, WORD wEjectType);	ErrorCode SCNMICRCancelFunction(MfEjectType)
int BiSCNSelectScanFace(int iHandle, BYTE bFace);	ErrorCode SCNSelectScanFace(ScanSide)
int BiGetPrnCapability(int, BYTE, LPBYTE, LPBYTE);	ErrorCode GetPrnCapability(Byte, out Byte()&)
	ErrorCode GetPrnCapability(Byte, out String&)
int BiCloseMonPrinter(int);	ErrorCode CloseMonPrinter()
int BiGetRealStatus(int, LPDWORD);	ErrorCode GetRealStatus(out ASB&)
int BiSCNMICRFunctionContinuously(int iHandle, LPVOID lpvStruct, WORD wFunction);	ErrorCode SCNMICRFunctionContinuously (FunctionType)
	ErrorCode SCNMICRFunctionContinuously (MF, FunctionType)
int BiSCNMICRFunctionPostPrint (int iHandle, LPVOID lpvStruct, WORD wFunction);	ErrorCode SCNMICRFunctionPostPrint(FunctionType)
	ErrorCode SCNMICRFunctionPostPrint (MF, FunctionType)
int BiSCNMICRSetStatusBackFunction (int, int (CALLBACK EXPORT *pScnMicrCB) (DWORD dwTransactionNumber, WORD wMainStatus, WORD wSubStatus, LPSTR lpPortName));	ErrorCode SCNMICRSetStatusBack()
	SCNMICRStatusCallback
int BiSCNMICRSetStatusBackWnd (int, long, LPDWORD, LPWORD, LPWORD);	-
int BiSCNMICRCancelStatusBack(int);	ErrorCode SCNMICRCancelStatusBack()
int BiSetNumberOfDocuments(int, BYTE bNumber);	ErrorCode SetNumberOfDocuments(Byte)
int BiGetMicrText(int, DWORD, LPMF_MICR);	ErrorCode GetMicrText(Int32, MFMicr)
int BiMICRClearSpaces(int, BYTE bClearSpace);	ErrorCode MICRClearSpaces(RemoveSpace)
int BiSetOcrABAreaOrigin(int nHandle, BYTE bOrigin);	ErrorCode SetOcrABAreaOrigin(OcrOrigin)
int BiGetOcrABText (int, DWORD, BYTE, LPCSTR, LPMF_OCR_AB);	ErrorCode GetOcrABText (UInt32, OcrImageSource, String, MFOcrAb)
int BiGetScanImage(int, DWORD, LPMF_SCAN);	ErrorCode GetScanImage(Int32, MFScan)
int BiGetBarcodeData(int, DWORD, LPMF_BARCODE);	ErrorCode GetBarcodeData(Int32, MFBarcode)
int BiDecodeBarcode(int, LPCSTR, LPMF_BARCODE);	ErrorCode DecodeBarcode(String, MFBarcode)

Win32 API	.NET API
int BiDecodeBarcodeMemory(int, LPBYTE, LPMF_BARCODE);	ErrorCode DecodeBarcodeMemory(Bitmap, MFBarcode)
int BiGetTransactionNumber(int, LPDWORD);	ErrorCode GetTransactionNumber(out Int32&)
	Int32 TransactionNumber
int BiSetTransactionNumber(int, DWORD);	ErrorCode SetTransactionNumber(Int32)
	Int32 TransactionNumber
int BiGetPrintStation(int, LPWORD);	ErrorCode GetPrintStation(out PrintingStation&)
int BiSetPrintStation(int, WORD);	ErrorCode SetPrintStation(PrintingStation)
int BiPrintText(int, LPSTR, MF_DECORATE);	ErrorCode PrintText(String, MFDecorate)
int BiPrintImage(int, LPSTR);	ErrorCode PrintImage(String)
int BiPrintMemoryImage(int, LPBYTE, DWORD);	ErrorCode PrintMemoryImage(Bitmap)
int BiGetPrintSize(int, LPWORD, LPWORD);	ErrorCode GetPrintSize(out Int32&, out Int32&)
	Size PrintSize
int BiSetPrintSize(int, WORD, WORD);	ErrorCode SetPrintSize(Int32, Int32)
	Size PrintSize
int BiGetPrintPosition(int, LPWORD, LPWORD);	ErrorCode GetPrintPosition(out Int32&, out Int32&)
	Point PrintPosition
int BiSetPrintPosition(int, WORD, WORD);	ErrorCode SetPrintPosition(Int32, Int32)
	Point PrintPosition
int BiSetEndorseDirection(int, BYTE bDirection);	ErrorCode SetEndorseDirection (ElectricEndorseDirection)
int BiUpdateEndorseText (int, LPSTR(3), DWORD(3), WORD(3), WORD(3));	ErrorCode UpdateEndorseText (String(), Int32(), FontType(), FontScale())
	ErrorCode UpdateEndorseText(MFPrint)
int BiBufferedPrint(int, DWORD);	ErrorCode BufferedPrint(PrintBuffer)
int BiSetTransactionNumberWithIncremental (int, DWORD, DWORD);	ErrorCode SetTransactionNumberWithIncrement (Int32, Int32)
	Int32 TransactionIncrement
int BiSetBehaviorToScnResult(int, BYTE, BYTE, BYTE);	ErrorCode SetBehaviorToScnResult (MfEject, MfStamp, MfProcessContinue)
int BiSetPaperThickness(int, WORD);	ErrorCode SetPaperThickness(UInt16)
int BiRingBuzzer(int, BYTE, BYTE, WORD, WORD);	ErrorCode RingBuzzer (MfBuzzerTone, Byte, UInt16, UInt16)
int BiSetWaterfallMode(int, BYTE bWaterfallMode);	ErrorCode SetWaterfallMode(WaterfallMode)
int BiGetIQAResult(int, DWORD dwTransactionNumber, LPMF_IQA_RESULT);	ErrorCode GetIqaResult(int, MFIqaResult)
int BiGetVersion(int, int, LPVERSION_INFO);	ErrorCode GetVersion (DriverType, VersionType, MFVersion)

Win32 API	**.NET API**
int BiESCNEnable(BYTE);	ErrorCode ESCNEnable(Storage)
int BiESCNGetAutoSize(int, LPBYTE);	ErrorCode ESCNGetAutoSize(out AutoSize&)
	AutoSize ESCNAutoSize
int BiESCNSetAutoSize(int, BYTE);	ErrorCode ESCNSetAutoSize(AutoSize)
	AutoSize ESCNAutoSize
int BiESCNGetCutSize(int, LPWORD);	ErrorCode ESCNGetCutSize(out Int32&)
	Int32 ESCNCutSize
int BiESCNSetCutSize(int, WORD);	ErrorCode ESCNSetCutSize(Int32)
	Int32 ESCNCutSize
int BiESCNGetRotate(int, LPBYTE);	ErrorCode ESCNGetRotate(out Rotate&)
	Rotate ESCNRotate
int BiESCNSetRotate(int, BYTE);	ErrorCode ESCNSetRotate(Rotate)
	Rotate ESCNRotate
int BiESCNGetDocumentSize(int, LPWORD, LPWORD);	ErrorCode ESCNGetDocumentSize (out Int32&, out Int32&)
	Size ESCNDocumentSize
int BiESCNSetDocumentSize(int, WORD, WORD);	ErrorCode ESCNSetDocumentSize(Int32, Int32)
	Size ESCNDocumentSize
int BiESCNGetDeSkew(int, LPWORD);	ErrorCode ESCNGetDeSkew(out UInt16&)
	UInt16 ESCNDeSkew
int BiESCNGetDeSkew(int, LPWORD);	ErrorCode ESCNSetDeSkew(UInt16)
	UInt16 ESCNDeSkew
int BiESCNDefineCropArea (int, BYTE, WORD, WORD, WORD, WORD);	ErrorCode ESCNDefineCropArea (Byte, Int32, Int32, Int32, Int32)
	ErrorCode ESCNDefineCropArea(Byte, Rectangle)
int BiESCNGetMaxCropAreas(int, LPBYTE);	ErrorCode ESCNGetMaxCropAreas(out Int32&)
int BiESCNStoreImage(int, DWORD, LPSTR, LPSTR, BYTE);	ErrorCode ESCNStoreImage(Int32, String, String, Byte)
int BiESCNRetrieveImage (int, DWORD, LPSTR, LPSTR, LPDWORD, LPBYTE);	ErrorCode ESCNRetrieveImage (Int32, String, String, out Byte()&)
	ErrorCode ESCNRetrieveImage (Int32, String, String, out Image&)
int BiESCNClearImage(int, BYTE, DWORD, LPSTR, LPSTR);	ErrorCode ESCNClearImage (ClearImageFlag, Int32, String, String)
int BiESCNGetRemainingImages(int, LPBYTE);	ErrorCode ESCNGetRemainingImages(out Int32&)
	Int32 ESCNRemainingImages
-	ErrorCode LastError
-	Boolean IsValid
-	MfDone

Class list

The TM-S1000 Driver implements .NET Wrapper. (The TM-S1000 is controlled with .NET Wrapper from your .NET applications.)
The .NET Wrapper has the following classes.

Class	Description
MFDevice	Main class. The device is controlled by calling a property method defined in this class.
MFBase	Class corresponding to the MF_BASE01 structure in Win32Wrapper.
MFMicr	Class corresponding to the MF_MICR structure in Win32Wrapper.
MFScan	Class corresponding to the MF_SCAN structure in Win32Wrapper.
MFPrint	Class corresponding to the MF_PRINT01 structure in Win32Wrapper.
MFTrueType	Class used for the PrintText method.
MFProcess	Class corresponding to the MF_PROCESS structure in Win32Wrapper.
MFIqa	Class corresponding to the MF_IQA structure in Win32Wrapper.
MFIqaResult	Class corresponding to the MF_IQA_RESULT structure in Win32Wrapper.
MFBarcode	Class corresponding to the MF_BARCODE structure in Win32Wrapper.
MFOcrAb	Class corresponding to the MF_OCR_AB structure in Win32Wrapper.
MFVersion	Class corresponding to the VERSION_INFO structure in Win32Wrapper.

Constant number list

.NET Wrapper has the following constant numbers.

Type of enumeration	Element	API used
ASB	ASB_NO_RESPONSE	StatusCallbackHandler, StatusCallbackHandlerEx, MFDevice class Status Property
	ASB_OFF_LINE	
	ASB_COVER_OPEN	
	ASB_WAIT_PEPRT_EJECT	
	ASB_MECHANICAL_ERR	
	ASB_UNRECOVER_ERR	
	ASB_PAPER_INTERMEDIATE	
	ASB_MAIN_NEAR_FULL	
	ASB_EJECT_SENSOR_NO_PAPER	
	ASB_SUB_NEAR_FULL	
	ASB_SLIP_PAPER_SIZE	
	ASB_ASF_PAPER	
	ASB_STAMP_EXIST	
	ASB_WAIT_INSERT	
	ASB_FRANKING_SENSOR	

Type of enumeration	Element	API used
AutoSize	CROP_AUTOSIZE_ENABLE	MFDevice class AutoSize Property, ESCNGetAutoSize method, ESCNSetAutoSize method
	CROP_AUTOSIZE_DISABLE	
CharSelect	MF_CONTINUE_ANALYSIS	MICRSelectDataHandling method
	MF_INTERRUPT_ANALYSIS	
Check	CHECK_PERSONAL	MFMicr class CheckType Property
	CHECK_BUSINESS	
	CHECK_UNKNOWN	
ClearImageFlag	CROP_CLEAR_ALL_IMAGE	ESCNClearImage method
	CROP_CLEAR_BY_FILEID	
	CROP_CLEAR_BY_FILEINDEX	
	CROP_CLEAR_BY_IMAGETAGDATA	
Color	EPS_BI_SCN_MONOCHROME	SCNSetImageQuality method
ColorDepth	EPS_BI_SCN_1BIT	SCNSetImageQuality method
	EPS_BI_SCN_8BIT	
CounterIndex	CHECK_READ	GetCounter method, ResetCounter method
	CHECK_SCAN	
	OPERATION_TIME	
	HOPPER_SWITCH	
	FRANKING	
	POCKET_CHANGE	
Country	COUNTRY_USA	MFMicr class CountryCode Property
	COUNTRY_CANADA	
	COUNTRY_MEXICO	
	COUNTRY_UNKNOWN	
DetailSelect	MF_DETAILINFO_NOTADDED	MICRSelectDataHandling method
	MF_DETAILINFO_ADDED	
DriverType	DRIVER_TYPE_J9000	GetVersion method
	DRIVER_TYPE_S1000	
EndorsementLine	MF_PRINT_LINE_1	UpdateEndorseText method, MFPrint class String Property, FontType Property, FontScale Property
	MF_PRINT_LINE_2	
	MF_PRINT_LINE_3	
EndorsementType	MF_PRINT_TYPE_ELECTRIC_ENDORSE_ONLY	MFPrint class ElectricEndorse Property
	MF_PRINT_TYPE_ELECTRIC_ENDORSE_EXTEND	

Type of enumeration	Element	API used
ErrorCode	SUCCESS	All method
	ERR_TYPE	
	ERR_OPENED	
	ERR_NO_PRINTER	
	ERR_NO_TARGET	
	ERR_NO_MEMORY	
	ERR_HANDLE	
	ERR_TIMEOUT	
	ERR_ACCESS	
	ERR_PARAM	
	ERR_NOT_SUPPORT	
	ERR_NOT_EPSON	
	ERR_WITHOUT_CB	
	ERR_BUFFER_OVER_FLOW	
	ERR_ENABLE	
	ERR_DISK_FULL	
	ERR_NO_IMAGE	
	ERR_ENTRY_OVER	
	ERR_CROPAREAID	
	ERR_EXIST	
	ERR_NOT_FOUND	
	ERR_IMAGE_FILEOPEN	
	ERR_IMAGE_UNKNOWNFORMAT	
	ERR_IMAGE_FAILED	
	ERR_IMAGE_FILEREAD	
	ERR_PAPERINSERT_TIMEOUT	
	ERR_EXEC_FUNCTION	
	ERR_EXEC_MICR	
	ERR_EXEC_SCAN	
	ERR_RESET	
	ERR_ABORT	
	ERR_MICR	
	ERR_SCAN	
	ERR_LINE_OVERFLOW	
	ERR_NOT_EXEC	
	ERR_SIZE	

Type of enumeration	Element	API used
ErrorCode	ERR_PAPER_PILED	All method
	ERR_PAPER_JAM	
	ERR_COVER_OPEN	
	ERR_MICR_BADDATA	
	ERR_MICR_PARSE	
	ERR_MICR_NOISE	
	ERR_SCN_COMPRESS	
	ERR_PAPER_EXIST	
	ERR_PAPER_INSERT	
	ERR_SCAN_IQA	
	ERR_BARCODE_NODATA	
ErrorSelect	ES_STOP_ALL	MICRSelectDataHandling method
	ES_CONTINUE_ALL	
	ES_CONTINUE_DOUBLEFEED	
	ES_CONTINUE_NODATA	
	ES_CONTINUE_BADDATA	
	ES_CONTINUE_NOISE	
ExOption	EPS_BI_SCN_MANUAL	SCNSetImageQuality method
	EPS_BI_SCN_SHARP	
	EPS_BI_SCN_SHARP_CUSTOM	
	EPS_BI_SCN_SHARP_CUSTOM2	
FontScale	MF_PRINT_FONT_W1_H1	MFPrint class FontSize Property UpdateEndorseText method
	MF_PRINT_FONT_W1_H2	
	MF_PRINT_FONT_W2_H1	
	MF_PRINT_FONT_W2_H2	
FontType	MF_PRINT_FONT_A	
	MF_PRINT_FONT_B	
Format	EPS_BI_SCN_TIFF	SCNSetImageFormat method, SCNGetImageFormat method
	EPS_BI_SCN_RASTER	
	EPS_BI_SCN_BITMAP	
	EPS_BI_SCN_TIFF256	
	EPS_BI_SCN_JPEGHIGH	
	EPS_BI_SCN_JPEGNORMAL	
	EPS_BI_SCN_JPEGLOW	
	EPS_BI_SCN_JTIFF	

Type of enumeration	Element	API used
FunctionType	MF_EXEC	SCNMICRFunction method, SCNMICRFunctionPostPrint method, SCNMICRFunctionContinuously method
	MF_CONTINUE	
	MF_MICR_RETRANS	
	MF_SCAN_FRONT_RETRANS	
	MF_SCAN_BACK_RETRANS	
	MF_SET_BASE_PARAM	
	MF_SET_MICR_PARAM	
	MF_SET_SCAN_PARAM	
	MF_SET_SCAN_FRONT_PARAM	
	MF_SET_SCAN_BACK_PARAM	
	MF_SET_BARCODE_PARAM	
	MF_SET_BARCODE_FRONT_PARAM	
	MF_SET_BARCODE_BACK_PARAM	
	MF_SET_PRINT_PARAM	
	MF_SET_PROCESS_PARAM	
	MF_CLEAR_BASE_PARAM	
	MF_CLEAR_MICR_PARAM	
	MF_CLEAR_SCAN_PARAM	
	MF_CLEAR_SCAN_FRONT_PARAM	
	MF_CLEAR_SCAN_BACK_PARAM	
	MF_CLEAR_BARCODE_PARAM	
	MF_CLEAR_BARCODE_FRONT_PARAM	
	MF_CLEAR_BARCODE_BACK_PARAM	
	MF_CLEAR_PRINT_PARAM	
	MF_CLEAR_PROCESS_PARAM	
	MF_GET_BASE_DEFAULT	
	MF_GET_MICR_DEFAULT	
	MF_GET_SCAN_DEFAULT	
	MF_GET_SCAN_FRONT_DEFAULT	
	MF_GET_SCAN_BACK_DEFAULT	
	MF_GET_BARCODE_DEFAULT	
	MF_GET_BARCODE_FRONT_DEFAULT	
	MF_GET_BARCODE_BACK_DEFAULT	
	MF_GET_PRINT_DEFAULT	
	MF_GET_PROCESS_DEFAULT	

Type of enumeration	Element	API used
MainStatus	MF_FUNCTION_START	SCNMICRStatusCallbackHandler
	MF_CHECKPAPER_PROCESS_START	
	MF_DATARECEIVE_START	
	MF_DATARECEIVE_DONE	
	MF_CHECKPAPER_PROCESS_DONE	
	MF_FUNCTION_DONE	
	MF_ERROR_OCCURED	
MfActivateMode	MF_ACTIVATE_MODE_CONFIRMATION	MFProcess class ActivationMode Property
	MF_ACTIVATE_MODE_HIGH_SPEED	
MfBuzzerCount	MF_BUZZER_DISABLE	MFBase class BuzzerCount Property
	MF_BUZZER_COUNT_ONE	
	MF_BUZZER_COUNT_TWO	
	MF_BUZZER_COUNT_MAX	
MfBuzzerHz	MF_BUZZER_HZ_4000	MFBase class BuzzerHz Property
	MF_BUZZER_HZ_440	
	MF_BUZZER_HZ_880	
MfBuzzerTone	MF_BUZZER_TONE_HIGH	RingBuzzer method
	MF_BUZZER_TONE_MIDDLE	
	MF_BUZZER_TONE_LOW	
MfBuzzerType	MF_BUZZER_TYPE_SUCCESS	MFBase class BuzzerHz Property BuzzerCount Property
	MF_BUZZER_TYPE_ERROR	
	MF_BUZZER_TYPE_WFEED	
MfCancel	MF_CANCEL_DISABLE	MFProcess class Cancel Property MFIqa class Cancel Property MFBarcode class Cancel Property
	MF_CANCEL_ENABLE	
MfDirection	MF_OCR_LEFTRIGHT	MFOcrAb class Direction Property
	MF_OCR_TOPBOTTOM	
	MF_OCR_RIGHTLEFT	
	MF_OCR_BOTTOMTOP	
MfEject	MF_EJECT_MAIN_POCKET	SetBehaviorToScnResult method, MFProcess class Eject Property MFIqa class Eject Property MFBarcode class Eject Property
	MF_EJECT_SUB_POCKET	
	MF_EJECT_NOEJECT	

Type of enumeration	Element	API used
MfEjectType	MF_EJECT_DISCHARGE	MFBase class SuccessEject Property, ErrorEject Property
	MF_EJECT_RELEASE	
	MF_EXIT_ERROR_DISCHARGE	
	MF_EXIT_ERROR_RELEASE	
MfErrorSelect	MF_ERROR_SELECT_NODETECT	MFProcess class Select Property MFIqa class ErrorSelect Property MFBarcode class ErrorSelect Property
	MF_ERROR_SELECT_DETECT	
MfMicrFont	MF_MICR_FONT_E13B	MFMicr class Font Property
	MF_MICR_FONT_CMC7	
MfMicrType	MF_MICR_USE_MICR	MFMicr class MicrOcrSelect Property
	MF_MICR_USE_OCR	
	MF_MICR_USE_BOTH	
MfNearFullSelect	MF_NEARFULL_NOT_PERMIT	MFProcess class NearFullSelect Property
	MF_NEARFULL_MAIN_PERMIT	
	MF_NEARFULL_SUB_PERMIT	
	MF_NEARFULL_PERMIT	
MfNVMemory	MF_BASE_NVMEMORY_NOT_USE	MFBase class UseNVMemory Property
	MF_BASE_NVMEMORY_USE	
MfOcrType	MF_OCR_FONT_OCRA_NUM	MFOcrAb class OcrType Property
	MF_OCR_FONT_OCRA_ALPHA	
	MF_OCR_FONT_OCRA_ALPHANUM	
	MF_OCR_FONT_OCRA_ALPHANUM_WOOH	
	MF_OCR_FONT_OCRA_ALPHANUM_WOZERO	
	MF_OCR_FONT_OCRB_NUM	
	MF_OCR_FONT_OCRB_ALPHA	
	MF_OCR_FONT_OCRB_ALPHANUM	
	MF_OCR_FONT_OCRB_ALPHANUM_WOOH	
	MF_OCR_FONT_OCRB_ALPHANUM_WOZERO	
MfPaperType	MF_PAPER_TYPE_OTHER	MFProcess class PaperType Property
	MF_PAPER_TYPE_CHECK	
MfProcessContinue	MF_PROCESS_CONTINUE_OVERLAP	SetBehaviorToScnResult method
	MF_PROCESS_CONTINUE_NOOVERLAP	
	MF_PROCESS_CONTINUE_CANCEL	
MfResult	MF_RESULT_NONE	MFProcess class ResultPartialData Property
	MF_RESULT_PARTIAL	

Type of enumeration	Element	API used
MfScanDpi	MF_SCAN_DPI_DEFAULT	MFScan class Resolution Property MFBarcode class Resolution Property
	MF_SCAN_DPI_100	
	MF_SCAN_DPI_120	
	MF_SCAN_DPI_200	
	MF_SCAN_DPI_240	
	MF_SCAN_DPI_H240_V200	
MfSpeceHandling	OCR_SPACE_ENABLE	MFOcrAB class SpeceHandling Property
	OCR_SPACE_DISABLE	
MfStamp	MF_STAMP_DISABLE	MFProcess class Stamp Property
	MF_STAMP_ENABLE	
MFPrintAttributes	MF_PRINT_NO_ATTRIBUTE	MFPrint class Attribute Property
	MF_PRINT_BOLD	
	MF_PRINT_UNDERLINE_1	
	MF_PRINT_UNDERLINE_2	
	MF_PRINT_BLACK	
	MF_PRINT_1ST_COLOR	
	MF_PRINT_COLOR	
	MF_PRINT_2ND_COLOR	
	MF_PRINT_MIXED	
	MF_PRINT_REVERSEVIDEO	
OcrImageSource	OCR_SOURCE_TRANSACTION_NUMBER	GetOcrABText method
	OCR_SOURCE_IMAGE_FILE	
OpenType	TYPE_PORT	OpenMonPrinter method, MFDevice constructor
	TYPE_PRINTER	
PrintBuffer	MF_PRT_BUFFERING	BufferedPrint method
	MF_PRT_EXEC	
	MF_PRT_CLEAR	
PrintColor	MF_PRINT_DEFAULT	MFTrueType class Color Property
	MF_PRINT_BLACK	
	MF_PRINT_COLOR	
	MF_PRINT_MIXED	
PrintDirection	MF_PRINT_DIRECTION_DOUBLE	MFPrint class Dicrection Property
	MF_PRINT_DIRECTION_SINGLE	
PrintingStation	MF_ST_ROLLPAPER	GetPrintStation method, SetPrintStation method, MFDevice class PrintStation Property
	MF_ST_VALIDATION	
	MF_ST_E_ENDORSEMENT	

Type of enumeration	Element	API used
PrintSpeed	MF_PRINT_SPEED_NORMAL	MFPrint class Speed Property
	MF_PRINT_SPEED_HIGH	
	MF_PRINT_SPEED_ECONOMY	
Rotate	CROP_ROTATE_ENABLE	ESCNGetRotate method, ESCNSetRotate method, MFDevice class Rotate Property
	CROP_ROTATE_DISABLE	
ScanSide	MF_SCAN_FACE_FRONT	SCNSelectScanFace method
	MF_SCAN_FACE_BACK	
ScanUnit	EPS_BI_SCN_UNIT_CHECKPAPER	SCNSelectScanUnit method
	EPS_BI_SCN_UNIT_CARD	
Storage	CROP_STORE_MEMORY	ESCNEnable method
	CROP_STORE_FILE	
VersionType	VERSION_TYPE_DRIVER	GetVersion method
	VERSION_TYPE_USB	
	VERSION_TYPE_MICR	
	VERSION_TYPE_MICR_E13B	
	VERSION_TYPE_MICR_CMC7	
	VERSION_TYPE_OCR	
	VERSION_TYPE_IMAGE	
	VERSION_TYPE_IQA	
	VERSION_TYPE_BARCODE	
ClearSpace	CLEAR_SPACE_DISABLE	MICRClearSpaces method
	CLEAR_SPACE_ENABLE	
OCROrigin	OCR_ORIGIN_TOP_LEFT	SetOcrABAreaOrigin method
	OCR_ORIGIN_BOTTOM_LEFT	
	OCR_ORIGIN_TOP_RIGHT	
	OCR_ORIGIN_BOTTOM_RIGHT	
EndorseDirection	EENDORSE_DIRECTION_LEFTRIGHT	SetEndorseDirection method
	EENDORSE_DIRECTION_TOPBOTTOM	
	EENDORSE_DIRECTION_RIGHTLEFT	
	EENDORSE_DIRECTION_BOTTOMTOP	
Waterfall	WATERFALL_MODE_DISABLE	SetWaterfallMode method
	WATERFALL_MODE_STANDARD	
	WATERFALL_MODE_INHERIT_POCKET	

Type of enumeration	Element	API used
IQA Test	MF_IQA_TEST_DISABLE	MFIqa class
	MF_IQA_TEST_ENABLE	
IQA Result	IQARESULT_NOT_TESTED	MFIqaResult class
	IQARESULT_PASS	
	IQARESULT_NOT_PASS	
MfBarcodeTargetColor	MF_BARCODE_TARGET_COLOR_GRAY	MFBarcode class TargetColor Property
MfBarcodeSymbol	MF_BARCODE_SYMBOL_CODABAR	MFBarcode class BarcodeInfo.SymbolMask Property BarcodeData.Symbol Property
	MF_BARCODE_SYMBOL_CODE128	
	MF_BARCODE_SYMBOL_CODE39	
	MF_BARCODE_SYMBOL_ITF	
	MF_BARCODE_SYMBOL_EAN_JAN	
	MF_BARCODE_SYMBOL_UPC_A	
	MF_BARCODE_SYMBOL_UPC_E	
MfBarcodeDirection	MF_BARCODE_DIRECTION_ALL	MFBarcode class BarcodeInfo.Direction Property BarcodeData.Direction Property
	MF_BARCODE_DIRECTION_LEFTRIGHT	
	MF_BARCODE_DIRECTION_TOPBOTTOM	
	MF_BARCODE_DIRECTION_RIGHTLEFT	
	MF_BARCODE_DIRECTION_BOTTOMTOP	

Chapter 4
Reference

This chapter describes the TM-S1000 .NET API and syntax.

Device information

Device Status

asb	ON/OFF	Value	Status	Response
ASB_NO_RESPONSE	ON	0x00000001	No Device response	• Check that the device is powered on, and that the cable is connected. • Check the specified device name, and also the computer port.
	OFF	0x00000000	Device Responds	-
ASB_OFF_LINE	ON	0x00000008	Off-line	Check the other ASBs to eliminate them as a cause for the device being offline.
	OFF	0x00000000	On-line	-
ASB_COVER_OPEN	ON	0x00000020	Cover is open.	Close the cover.
	OFF	0x00000000	Cover is closed.	-
ASB_WAIT_PEPRT_EJECT	ON	0x00000100	There is a check paper in the transport path.	Remove the paper as described in the user's manual.
	OFF	0x00000000	-	-
ASB_MECHANICAL_ERR	ON	0x00000400	Recoverable Errors generated Refer to Technical Reference Guide	Check the other ASBs, eliminate the source of any errors, and then either power-on the scanner again or issue an error cancel (CancelError) command.
	OFF	0x00000000	No recoverable error.	-
ASB_UNRECOVER_ERR	ON	0x00002000	Unrecoverable Errors generated Refer to Technical Reference Guide	Immediately turn off the power to the device.
	OFF	0x00000000	No unrecoverable error.	-
ASB_PAPER_INTERMEDIATE	ON	0x00010000	The intermediate sensor senses no paper.	-
	OFF	0x00000000	The intermediate sensor senses that there is paper.	Remove the paper as described in the user's manual.
ASB_MAIN_NEAR_FULL	ON	0x00020000	The main pocket is not nearly full.	-
	OFF	0x00000000	The main pocket is nearly full.	Remove the paper from the main pocket.

asb	ON/OFF	Value	Status	Response
ASB_EJECT_SENSOR_NO_PAPER	ON	0x00040000	The eject sensor senses no paper.	-
	OFF	0x00000000	The eject sensor senses that there is paper.	Remove the paper as described in the user's manual.
ASB_SUB_NEAR_FULL	ON	0x00080000	The sub pocket is not nearly full.	-
	OFF	0x00000000	The sub pocket is nearly full.	Remove the paper from the sub-pocket.
ASB_SLIP_PAPER_SIZE	ON	0x00200000	No check paper at the paper length sensor.	-
	OFF	0x00000000	Check paper at the paper length sensor.	Remove the paper as described in the user's manual.
ASB_ASF_PAPER	ON	0x00400000	There is no paper in the ASF.	-
	OFF	0x00000000	There is paper in the ASF.	-
ASB_STAMP_EXIST	ON	0x02000000	The franking cartridge is not installed.	When ASB_STAMP_EXIST = NO and the Franker is enable, the application needs to prompt a warning.
	OFF	0x00000000	The franking cartridge is installed.	-
ASB_WAIT_INSERT	ON	0x20000000	Waiting for paper.	-
	OFF	0x00000000	-	-
ASB_FRANKING_SENSOR	ON	0x40000000	The franking sensor senses no paper.	-
	OFF	0x00000000	The franking sensor senses there is paper.	Remove the paper as described in the user's manual.

Maintenance Counter

Counter Number (read no)	Resetability	Counter (readcounter)	(Unit)
60 (3CH)	Resetable	Magnetic text read count	(No of times)
61 (3DH)	Resetable	Check paper image read count	(No of times)
70 (46H)	Resetable	Product operating time	(Hours)
80 (50H)	Resetable	Hopper open/close count	(No of times)
81 (51H)	Resetable	Franking count	(No of times)
82 (52H)	Resetable	Pocket switching count	(No of times)
188 (BCH)	Cumulative	Magnetic text read count	(No of times)
189 (BDH)	Cumulative	Check paper image read count	(No of times)
198 (C6H)	Cumulative	Product operating time	(Hours)
208 (D0H)	Cumulative	Hopper open/close count	(No of times)
209 (D1H)	Cumulative	Franking count	(No of times)
210 (D2H)	Cumulative	Pocket switching count	(No of times)

Type ID

Bit	ON/OFF	Value	Function
0	-	0x00	-
1	-	0x00	-
2	-	0x00	-
3	ON	0x08	MICR reader is installed. (Fixed to 1)
4	-	0x00	Fixed to 0
5	-	0x00	-
6	-	0x00	-
7	-	0x00	Fixed to 0

Device ID

Device ID (Set to *prnID*)	Type of Device ID	Data to obtain
1	Product ID	1-byte data. Product ID is 111(6FH).
2	Type ID	1-byte data. See Table (A) for the data to obtain.
33	Type information (1)	2-byte data. See Table (B) for the data to obtain.
65	Version of firmware	String data. The obtained data varies depending on the firmware version. Example: "1.00 ESC/POS"
66	Manufacture name	String data. Default is "EPSON". Example: "EPSON"
67	Product name	String data. The obtained data varies depending on the prodct. Example: "TM-S1000"
68	Serial number	String data. The obtained data varies depending on the device. Example: "DEKK000015"
112	Type information (2)	1-byte data. See Table (C) for data to obtain.

(A) Type ID (Device ID = 2)

Bit	ON/OFF	Value	Function
0	-	0x00	-
1	-	0x00	-
2	-	0x00	-
3	ON	0x08	MICR reader is installed. (Fixed to 1)
4	-	0x00	-
5	-	0x00	-
6	-	0x00	-
7	-	0x00	-

(B) Type information (Device ID = 33)

First byte

Bit	ON/OFF	Value	Function
0	-	0x00	-
1	-	0x00	-
2	-	0x00	-
3	ON	0x08	MICR reader is installed. (Fixed to 1)
4	ON	0x10	Image reader is installed. (Fixed to 1)
5	-	0x00	-
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Second byte

Bit	ON/OFF	Value	Function
0	-	0x00	-
1	ON	0x02	Grayscale reading is supported. (Fixed to 1)
2	ON	0x04	Image reader is installed. (Fixed to 1)
3	ON	0x08	ASF is installed. (Fixed to 1)
4	-	0x00	-
5	-	0x00	-
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

(C) Type information (Device ID = 112)

Bit	ON/OFF	Value	Function
0	-	0x00	-
1	Model (Refer to "Model" on page 4-5)		
2			
3	-	0x00	-
4	ON	0x10	Waterfall is supported.
	OFF	0x00	Waterfall is unsupported.
5	OFF	0x00	2 pokets
6	-	0x40	Fixed to 1
7	-	0x00	-

Model

Model	1 Bit	2 Bit
30 dpm	OFF (0x00)	OFF (0x00)
60 dpm	ON (0x02)	OFF (0x00)
90 dpm	ON (0x02)	ON (0x04)

Offline Code (OfflineCode)

First Byte

Bit	ON/OFF	Value	Status
0	ON	0x02	CPU execution error generated
	OFF	0x00	CPU execution error not generated
1	ON	0x04	ROM error generated
	OFF	0x00	ROM error not generated
2	-	0x00	fixed to 0
3	-	0x00	fixed to 0
4	-	0x00	fixed to 0
5	-	0x00	fixed to 0
6	-	0x40	fixed to 1
7	-	0x00	fixed to 0

Second Byte

Bit	ON/OFF	Value	Status
0	ON	0x01	High voltage error generated
	OFF	0x00	High voltage error not generated
1	ON	0x02	Low voltage error generated
	OFF	0x00	Low voltage error not generated
2	-	0x00	fixed to 0
3	-	0x00	fixed to 0
4	-	0x00	fixed to 0
5	-	0x00	fixed to 0
6	-	0x40	fixed to 1
7	-	0x00	fixed to 0

Third Byte

Bit	ON/OFF	Value	Status
0	-	0x00	fixed to 0
1	-	0x00	fixed to 0
2	-	0x00	fixed to 0
3	-	0x00	fixed to 0
4	ON	0x04	CIS error generated
	OFF	0x00	CIS error not generated
5	-	0x00	fixed to 0
6	-	0x40	fixed to 1
7	-	0x00	fixed to 0

Fourth Byte

Bit	ON/OFF	Value	Status
0	-	0x00	fixed to 0
1	-	0x00	fixed to 0
2	-	0x00	fixed to 0
3	-	0x00	fixed to 0
4	-	0x00	fixed to 0
5	-	0x00	fixed to 0
6	-	0x40	fixed to 1
7	-	0x00	fixed to 0

Fifth Byte

Bit	ON/OFF	Value	Status
0	-	0x00	fixed to 0
1	-	0x00	fixed to 0
2	-	0x00	fixed to 0
3	-	0x00	fixed to 0
4	-	0x00	fixed to 0
5	-	0x00	fixed to 0
6	-	0x40	fixed to 1
7	-	0x00	fixed to 0

Offline Code (GetOfflineCodeByIndex)

First Byte (Recoverable Errors)

Bit	ON/OFF	Value	Status
0	-	0x00	fixed to 0
1	ON	0x02	CPU execution error generated
	OFF	0x00	CPU execution error not generated
2	ON	0x04	ROM error generated
	OFF	0x00	ROM error not generated
3	-	0x00	fixed to 0
4	-	0x00	fixed to 0
5	-	0x00	fixed to 0
6	-	0x40	fixed to 1
7	-	0x00	fixed to 0

Second Byte (Unrecoverable Errors)

Bit	ON/OFF	Value	Status
0	ON	0x01	High voltage error generated
	OFF	0x00	High voltage error not generated
1	ON	0x02	Low voltage error generated
	OFF	0x00	Low voltage error not generated
2	ON	0x04	CIS error generated
	OFF	0x00	CIS error not generated
3	-	0x00	fixed to 0
4	-	0x00	fixed to 0
5	-	0x00	fixed to 0
6	-	0x40	fixed to 1
7	-	0x00	fixed to 1

Third Byte (Recoverable Errors: mechanism position error)

Bit	ON/OFF	Value	Status
0	ON	0x01	Hopper position error generated
	OFF	0x00	Hopper position error not generate
1	ON	0x02	Stamp position error generated
	OFF	0x00	Stamp position error not generated
2	ON	0x04	Switching plate position error generated
	OFF	0x00	Switching plate position error not generated
3	-	0x00	fixed to 0
4	-	0x00	fixed to 0
5	-	0x00	fixed to 0
6	-	0x40	fixed to 1
7	-	0x00	fixed to 0

Fourth Byte (Recoverable Errors: paper jam error)

Bit	ON/OFF	Value	Status
0	ON	0x01	Cut paper present error generated
	OFF	0x00	Cut paper present error not generated
1	ON	0x02	Cut-sheet ejection error generated
	OFF	0x00	Cut-sheet ejection error not generated
2	ON	0x04	Cut paper length error generated
	OFF	0x00	Cut paper length error not generated
3	ON	0x08	Cut paper transport error generated
	OFF	0x00	Cut paper transport error not generated
4	-	0x00	fixed to 0
5	-	0x00	fixed to 0
6	-	0x40	fixed to 1
7	-	0x00	fixed to 0

Fifth Byte (Recoverable Errors: continuous read error that detection setting is available)

Bit	ON/OFF	Value	Status
0	ON	0x01	Cut paper double feed error generated
	OFF	0x00	Cut paper double feed error not generated
1	ON	0x02	Cut paper insertion direction error generated
	OFF	0x00	Cut paper insertion direction error not generated
2	ON	0x04	External noise error generated
	OFF	0x00	External noise error not generated
3	ON	0x08	Read error generated as a result of command indication
	OFF	0x00	Read error not generated as a result of command indication
4	-	0x00	fixed to 0
5	-	0x00	fixed to 0
6	-	0x40	fixed to 0
7	-	0x00	fixed to 0

MICR Status

Bit	ON/OFF	Value	Status
0	-	0x00	fixed to 0
1	-	0x02	fixed to 1
2	ON	0x04	MICR function non-selection
	OFF	0x00	MICR function selection
3	ON	0x08	Waiting for check sheet/cleaning sheet insertion
	OFF	0x00	-
4	-	0x10	fixed to 1
5	ON	0x20	TOF sensor senses no paper
	OFF	0x00	TOF sensor senses that there is paper
6	ON	0x40	BOF sensor senses no paper
	OFF	0x00	BOF sensor senses that there is paper
7	-	0x00	fixed to 0

TM-S1000 .NET API Error Handling

The following table shows how TM-S1000 .NET API handles each of the error return values.

ErrorCode	Description	Response
SUCCESS	Success	-
ERR_TYPE	nType parameter error	A constant value that is not defined is used in the header. Use a constant value that is defined.
ERR_OPENED	The specified device has already been opened	The port is already used by the other application. Close the application.
ERR_NO_PRINTER	The specified device driver does not exist	The power of the device is off, or the host is not connected. Check the device status.
ERR_NO_TARGET	An unsupported device was specified (The device's power is not On or the cable connections are faulty, etc.)	The current USB driver is not supported. Install the correct driver.
ERR_NO_MEMORY	Memory is insufficient	There is insufficient memory for running the driver. Close other applications or add more memory.
ERR_HANDLE	The handle value that specifies the device is incorrect	The handle value is incorrect. Check if the same handle value is used as the one when OpenMonPrinter is correctly finished.
ERR_TIMEOUT	A time out error occurred	API is not finished correctly. Check the device status.
ERR_ACCESS	Reading/writing with the device is not possible (printing in progress)	API is not finished correctly. Check the device status.
ERR_PARAM	Parameter error	This is a parameter error. Check if the set value is correct.
ERR_NOT_SUPPORT	Unsupported	This API function is not available. When using the scanner advanced function API, confirm that ESCNEnable is called at the beginning.
ERR_OFFLINE	It was opened in the offline state, so it cannot be used until the online state is recovered.	The device has gone offline. Check the device status.
ERR_NOT_EPSON	Cannot be used (device not EPSON)	The current device is not EPSON product. Use an EPSON's device.
ERR_WITHOUT_CB	SCNMICRFunctionPostPrint/ SCNMICRFunctionContinuously has not been run	• The callback destination is not registered on the driver. Call either SCNMICRSetStatusBackFunction or SCNMICRSetStatusBackWnd • The scanner is not in operation. Call it while the scanner is scanning.
ERR_BUFFER_OVER_FLOW	Buffer overflow error	The memory is insufficient for the required task. Add more memory.
ERR_ENABLE	Cannot be used because OpenMonPrinter is called	OpenMonPrinter is already called. Execute CloseMonPrinter, and then recall OpenMonPrinter.
ERR_DISK_FULL	There is insufficient free space on the disk	Disk capacity is insufficient. Review the operating environment.
ERR_NO_IMAGE	The image data does not exist	No image files are found. Check if the image file exists in the specified destination.

ErrorCode	Description	Response
ERR_CROPAREAID	The specified Crop Area does not exist	DefineCropArea is not configured. Set DefineCropArea, and then recall it.
ERR_EXIST	The specified data has already been saved	The parameter has already been registered. Change the parameter or call ESCNClearImage.
ERR_NOT_FOUND	No data error	• No data exist in the corresponding transaction number. Confirm that the transaction number is correct. • It is not registered yet. Check the parameter. • No corresponding module is found. Re-install the driver.
ERR_IMAGE_FILEOPEN	Open failure	Specified image file cannot be opened. Check if the other application is using it.
ERR_IMAGE_UNKNOWNFORMAT	Format injustice	File format is incorrect or not supported. Check if the image can be opened with the other application.
ERR_IMAGE_FAILED	Image data creation failed	Saving image file failed. Review the operating environment.
ERR_IMAGE_FILEREAD	Read of the image data file failed	Reading image file failed. Check if the file exists in the specified destination.
ERR_PAPERINSERT_TIMEOUT	Paper insertion time exceeded	Inserting the check sheet failed. Check if the check paper is correctly placed on ASF.
ERR_EXEC_FUNCTION	Cannot be used because the other API is being executed	The other API is in use. Retry after the other API is finished.
ERR_RESET	Cannot be used because the device is being reset	The device is being reset. Retry after the reset is finished.
ERR_ABORT	Canceled by SCNMICRCancelFunction	Reading operation is canceled. Determine whether to execute the operation again or not.
	Near full is detected	Notified when NearFullSelect of MFProcess class is set to other than MF_NEARFULL_PERMIT and Near full is detected during scanning.
ERR_MICR	Printer failed in MICR reading	MICR data is not read. Read it again.
ERR_SCAN	Printer failed in image scanning	Image data is not read. Read it again.
ERR_NOT_EXEC	Process not being executed	Reading operation is not executed. Read it again.
ERR_SIZE	Size excess error	SetPrintSize is not called, or the value is not valid. Execute it again after calling SetPrintSize.
ERR_PAPER_PILED	Paper pilling error	Double feed error is detected. Determine whether to execute the reading operation again or not.
ERR_PAPER_JAM	Paper jam has occurred	Paper jam error occurs. Remove the jammed paper and call CancelError.
ERR_COVER_OPEN	Cover open error	Path cover is opened. Close the cover.
ERR_MICR_NODATA	MICR data is not existing	MICR data does not exist. Determine whether to execute the reading operation again or not.

ErrorCode	Description	Response
ERR_MICR_BADDATA	MICR data is not able to recognize	MICR data is incorrect. Determine whether to execute the reading operation again or not.
ERR_MICR_PARSE	MICR data can not be parsed	Parsing operation of MICR data failed. Determine whether to execute the reading operation again or not.
ERR_MICR_NOISE	Noise error has occurred during MICR reading	External noise error is detected. Review the installation site, and determine whether to execute the reading operation again or not.
ERR_PAPER_EXIST	API can not be execute because there is a paper on the path	Paper exists on the paper feed path. Remove the paper.
ERR_PAPER_INSERT	Paper insertion error	Paper insertion error is detected. Check if the insertion direction is correct, and judge whether to execute the reading operation again or not.
ERR_LESS_CHECKS	The number of documents specified by SetNumberOfDocuments cannot be read.	Change the specified number of documents, or increase the number of documents to be read.
ERR_SCAN_IQA	Error has detected in the IQA test.	Call GetIQAResult to confirm the test result, and determine if reading should be tried again.
ERR_BARCODE_NODATA	Barcode cannot be detected.	• Check if the specifications of front/back side are correct. • Check if the specified barcode type and the decode direction are correct. • Read image data may be low in quality. Read it again then.

MFDevice Class

The MFDevice class is the primary interface with which to control the Epson multi-function device. The following Methods are provided by the driver's API.

Method	Description
MFDevice	Create Instance, open functionality supplied
ResetLastError	Clear the last error which occurred with this instance
OpenMonPrinter	Open connection to device
CloseMonPrinter	Close connection to device
GetRealStatus	Force communication with device to determine status
ResetPrinter	Force device to reset
CancelError	Clear error state of device
GetCounter	Read value from device internal maintenance counter
ResetCounter	Reset value in device internal maintenance counter
GetType	Get device type information
GetPrnCapability	Get device configuration
SetMonInterval	Configure monitoring interval for device communication
MICRSelectDataHandling	Set how MICR data is handled while scanning
MICRGetStatus	Get current MICR status
MICRCleaning	Clean device MICR head
SCNSelectScanFace	Set face of check to scan
SCNSelectScanUnit	Set which scanner to use for scanning
SCNSetImageQuality	Configure image quality settings
SCNGetImageQuality	Read image quality settings
SCNSetImageFormat	Configure image format for image transfer
SCNGetImageFormat	Read image format for image transfer
SCNSetScanArea	Configure scanning area
SCNGetScanArea	Read scanning area
SCNSetCroppingArea	Configure cropping area
SCNGetCroppingAreas	Read cropping area
SCNDeleteCroppingArea	Delete cropping area
ESCNEnable	Set ESCN method operation
ESCNSetAutoSize	Configure auto sizing
ESCNGetAutoSize	Read auto sizing state
ESCNSetCutSize	Configure image cut size
ESCNGetCutSize	Read image cut size value
ESCNSetRotate	Configure image rotation setting
ESCNGetRotate	Read image rotation setting
ESCNSetDocumentSize	Configure size of document to scan
ESCNGetDocumentSize	Read size of document that will be scanned

Method	Description
ESCNDefineCropArea	Configure cropping area
ESCNGetMaxCropArea	Determine maximum number of cropping areas
ESCNStoreImage	Store scanned image
ESCNRetrieveImage	Retreive scanned image
ESCNClearImage	Clear scanned image
ESCNGetRemainingImages	Read number of images that can be scanned
SCNMICRFunction	Configure and scan ID cards
SCNMICRFunctionContinuously	Configure and scan checks in high speed mode
SCNMICRFunctionPostPrint	Configure and scan checks with dynamic endorsement
SCNMICRCancelFunction	Cancel scanning operation
GetMicrText	Read check MICR and OCR information
GetScanImage	Read check scanned image
UpdateEndorseText	Update endoresement text
SetTransactionNumber	Configure the transaction number
GetTransactionNumber	Read the transaction number
SetTransactionNumberWithIncrement	Configure transaction number and it's increment rate
PrintText	Print text on roll paper or validation
PrintImage	Print image on roll paper or validation
SetPrintSize	Configure size to print image or barcode
GetPrintSize	Read size setting for image or barcode
SetPrintPosition	Configure location of where to print
GetPrintPosition	Read where next print will occur
SetPrintStation	Configure station where printing will occur
GetPrintStation	Determine where printing will occur
BufferedPrint	Start / stop / cancel buffered print command handling
SetStatusBack	Enable StatusCallback and StatusCallbackEx events
CancelStatusBack	Disable StatusCallback and StatusCallbackEx events
SCNMICRSetStatusBack	Enable SCNMICRStatusCallback event
SCNMICRCancelStatusBack	Disable SCNMICRStatusCallback event
SetBehaviorToScnResult	Specify an operation in Confirmation mode
RingBuzzer	Sound the internal buzzer
GetOfflineCodeByIndex	Acquire data that shows the device offline causes
GetOcrABText	Recognize OCR-A or OCR-B character string
PrintMemoryImage	Print an image
GetVersion	Acquire version information of the driver or module
ESCNGetDeSkew	Acquire a skew correction value
ESCNSetDeSkew	Set a skew correction value
SetNumberOfDocuments	Specify the number of documents to read

Method	Description
MICRClearSpaces	Clear spaces in MICR characters
SetOcrABAreaOrigin	Specify the origin of the OcrAB area
SetPaperThickness	Specify the double feed threshold
SetWaterfallMode	Specify the operation in Waterfall mode
SetEndorseDirection	Specify the direction of the Electric Endorse
GetIQAResult	Get IQA result
GetBarcodeData	Decode a barcode from the scanned image data, and obtain the result.
DecodeBarcode	Decode a barcode from the specified image file, and obtain the result.
DecodeBarcodeMemory	Decode the barcode from the image data on the application memory, and obtain the result.

The following Properties are exposed by the driver's API.

Property	Description
IsValid	(R) determine if instance has successfully opened device
LastError	(R) get last error
Status	(R) get most current device status
OfflineCode	(R) get reason for device offline state
ESCNAutoSize	(R/W) get/set auto sizing state
ESCNCutSize	(R/W) get/set cut size
ESCNRotate	(R/W) get/set image rotation state
ESCNDocumentSize	(R/W) get/set scan image size
ESCNDeSkew	(R/W) get/set skew angle
ESCNRemainingImages	(R) get remaining amount of crop data
TransactionNumber	(R/W) get/set transaction number
TransactionIncrement	(W) set transaction increment rate
PrintStation	(R/W) get/set station where printing will occur
PrintSize	(R/W) get/set print size of images and barcodes
PrintPosition	(R/W) get/set location where printing will occur

The following Events are exposed by the driver's API.

Event	Description
StatusCallback	Notify device status change
StatusCallbackEx	Notify device status change with port name
MfDone	Notify check scan state(SCNMICRFunction)
SCNMICRStatusCallback	Notify check scan state(SCNMICRFunctionPostPrint,SCNMICRFunctionContinuously)

MFDevice

Construct and instance of this class, connect to device if the extended version is used.

Syntax

MFDevice()

MFDevice(*OpenType type, String name*)

Argument

OpenType: Determine type of value supplied in name.

type	Description
TYPE_PORT	Port name is supplied (Example: "USB2")
TYPE_PRINTER	Device name is supplied (Example: "TM-S1000U")

name: Port name or device name to open.

Explanation

If the extended constructor is used methods other then OpenMonPrinter will be available for use.

The number of the device that is able to open at a time is to 32 units.

There are a limited number of methods that can be used if this method is called when device is offline.

Note

The handle value obtained is valid only in the same thread.

ResetLastError

Reset the last recorded error.

Syntax

void **ResetLastError**()

Explanation

After this method is called the LastError property will return SUCCESS until an error occurs.

OpenMonPrinter

This opens the device for which the status is being monitored.

Syntax

ErrorCode **OpenMonPrinter**(*OpenType type, String name*)

Argument

OpenType: Determine type of value supplied in name.

type	Description
TYPE_PORT	Port name is supplied (Example: "USB2")
TYPE_PRINTER	Device name is supplied (Example: "TM-S1000U")

name: Port name or device name to open.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_TYPE	Parameter error
ERR_OPENED	Device already opened or instance already open
ERR_NO_PRINTER	Specified device does not exist
ERR_NO_TARGET	Invalid device specified
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_TIMEOUT	Unable to complete within timeout
ERR_PARAM	Parameter error
ERR_NOT_EPSON	Connected device not an Epson device

Explanation

This method (or the extended constructor) is required before any other methods or parameters are used

CloseMonPrinter

Disconnect from the monitored device.

Syntax

ErrorCode **CloseMonPrinter**()

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly

Explanation

Disconnect and stop all communication with device. After this OpenMonPrinter must be called to reconnect to device and all callbacks must be re-enabled.

GetRealStatus

Force communication with device and read current status.

Syntax

ErrorCode **GetRealStatus**(*out ASB asb*)

Argument

asb: (out) Combination of device status bits.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

Since this method forces communication with device is it not suggested to use it within a polling loop. For polling use the Status property, for event based status monitoring use the StatusCallback or StatusCallbackEx events. Device Status, refer to "Device Status" on page 4-1.

Note

Note that when an error has occurred, if this argument is used frequently, the device buffer becomes full and ERR_ACCESS will occur.

ResetPrinter

Force the device to reset, reinitializing values.

Syntax

ErrorCode **ResetPrinter**()

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

This command forces the device to reset, all connections are reestablished automatically when the device returns to an online state.

CancelError

Resets the device's error state.

Syntax

ErrorCode **CancelError**()

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write

Explanation

If the device is in a error that from which it is possible to recover, the error is recovered and the error state is cleared.

GetCounter

Reads the value of a maintenance counter.

Syntax

ErrorCode **GetCounter**(*CounterIndex counter, bool cumulative, out int value*)

ErrorCode **GetCounter**(*byte counter, out int value*)

Argument

counter (byte): Value of the counter to read.
(Refer to "Maintenance Counter" on page 4-3)

counter (CounterIndex): Enumeration of the possible counters.
(Refer to "Maintenance Counter" on page 4-3)

cumulative: If true, read cumulative value.

value: (out) The value of the counter.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_OFFLINE	Device offline
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

Read the value of the maintenance counter. Valid counter values are listed in the refer to "Maintenance Counter" on page 4-3.

ResetCounter

Resets the maintenance counter.

Syntax

ErrorCode **ResetCounter**(*CounterIndex counter*)

ErrorCode **ResetCounter**(*byte counter*)

Argument

counter (byte): Value of the counter to read.
(Refer to "Maintenance Counter" on page 4-3)

counter (CounterIndex): Enumeration of the possible counters.
(Refer to "Maintenance Counter" on page 4-3)

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_OFFLINE	Device offline
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

This command will reset the value of any maintenance counters which can be reset, refer to "Maintenance Counter" on page 4-3.

GetType

Acquires the device's type ID.

Syntax

ErrorCode **GetType**(*out byte typeid, out byte font, out byte exrom, out byte euspecial*)

Argument

typeid: (out) The device's type id.

font: (out) The font installed on the device. (not used)

exrom: (out) Expanded flash usage. (not used)

euspecial: (out) The device's type id. (not used)

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_OFFLINE	Device offline
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

Typeid has bits defined as follows, refer to "Type ID" on page 4-3.

GetPrnCapability

Acquires device information specified by device ID.

Syntax

ErrorCode **GetPrnCapability**(*byte deviceID, out byte[] data*)

ErrorCode **GetPrnCapability**(*byte deviceID, out String data*)

Argument

deviceID: Specifies the device ID from which obtain information. See "Device ID" on page 4-4 for details on Device ID.

data (byte[]) (out) Byte array containing configuration information.

data (String): (out) String formatted capability.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_OFFLINE	Device offline
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

Device ID List, refer to "Device ID" on page 4-4.

Note

There are non-supported device IDs depending on models. If a non-supported device ID is specified, the error ERR_TIMEOUT will be returned.

SetMonInterval

Specifies the interval of the status information from the device expressed in milliseconds.
API that is compatible with the TM-J9000 API. No operation is possible with the TM-S1000 API.

Syntax

ErrorCode **SetMonInterval**(*int noPrnInterval, int prnInterval*)

Argument

noPrnInterval: Interval to use when not printing.

prnInterval: Interval to use when printing.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

This method sets the internal intervals used which printing and while not printing.

MICRSelectDataHandling

Selects the check reading operation.

Syntax

ErrorCode **MICRSelectDataHandling**(*CharSelect charSel, DetailSelect detailSel, ErrorSelect errorSel*)

Argument

CharSelect:

charSel	Description
MF_INTERRUPT_ANALYSIS	Stop analysis of misreading of a character
MF_CONTINUE_ANALYSIS	Insert for misread characters and continue processing

DetailSelect: Not Used

ErrorSelect:

errorSel	Description
ES_STOP_ALL	Processing is stopped with any error
ES_CONTINUE_ALL	Processing is continued irrelevant of error
ES_CONTINUE_DOUBLEFEED	Continue if double feed is detected
ES_CONTINUE_NODATA	Continue if data is not detected
ES_CONTINUE_BADDATA	Continue if invalid data is detected
ES_CONTINUE_NOISE	Continue if noise is detected in data

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

ES_CONTINUE_DOUBLEFEED, ES_CONTINUE_NODATA, ES_CONTINUE_BADDATA and ES_CONTINUE_NOISE may be combined to form a composite value.

Note

For compatibility with the TM-J9000/9100 driver, when MF_PROCESS structure is not set, and the initial value defined with the MF_PROCESS structure and the action defined with ErrorSelect are nonequivalent, the priority is given to the action defined with ErrorSelect.

MICRGetStatus

Acquire MICR status.

Syntax

ErrorCode **MICRGetStatus**(*out byte status*)

Argument

status: (out) Status of the MICR read.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_PARAM	Parameter error
ERR_OFFLINE	Device offline
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

The bits within status are defined as follows, refer to "MICR Status" on page 4-10.

MICRCleaning

Clean the MICR mechanism.

Syntax

ErrorCode **MICRCleaning**()

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_OFFLINE	Device offline

Explanation

If this function is called, the mechanism waits for the cleaning sheet to be inserted. Insert the cleaning sheet and carry out cleaning of the mechanism.

SCNSelectScanFace

Send and read raw data to and from the device.

Syntax

ErrorCode **SCNSelectScanFace**(*ScanSide scanFace*)

Argument

ScanSide:

scanFace	Description
MF_SCAN_FACE_FRONT	Select the front side of the check
MF_SCAN_FACE_BACK	Select the back side of the check

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

This method determines which scan face is affected by other methods and properties.

SCNSelectScanUnit

Send and read raw data to and from the device.

Syntax

ErrorCode **SCNSelectScanUnit**(*ScanUnit scanUnit*)

Argument

ScanUnit:

scanUnit	Description
EPS_BI_SCN_UNIT_CHECKPAPER	Select the check scanner
EPS_BI_SCN_UNIT_CARD	Select the ID card scanner

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

This method determines which scan unit is affected by other methods and properties.

Note

Default scanUnit is EPS_BI_SCN_UNIT_CHECKPAPER.

SCNSetImageQuality / SCNGetImageQuality

Set and get parameters pertaining to image quality.

Syntax

ErrorCode **SCNSetImageQuality**(*ColorDepth colorDepth, double threshold, Color color, ExOption exOption*)

ErrorCode **SCNGetImageQuality**(*out ColorDepth colorDepth, out double threshold, out Color color, out ExOption exOption*)

Argument

ColorDepth:

colorDepth	Description
EPS_BI_SCN_1BIT	Scan with one bit per pixel (black and white)
EPS_BI_SCN_8BIT	Scan with eight bits per pixel (grayscale)

threshold : Brightness threshold value (-1.0 ~ 1.0) for Black and White.
For auto detection is set as Double.NaN.

Color:

color	Description
EPS_BI_SCN_MONOCHROME	Scan monochrome image (not color)

ExOption:

exOption	Description
EPS_BI_SCN_MANUAL	Applies the value of bThreshold for Black and White.
EPS_BI_SCN_SHARP	Sharpness filter added
EPS_BI_SCN_SHARP_CUSTOM	Custom sharpness filter added
EPS_BI_SCN_SHARP_CUSTOM2	Second custom sharpness filter added
EPS_BI_SCN_SHARP_CUSTOM3	Edge-preserving smoothing Recommended when JPEG 2000 is used with your application.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

Use this method to configure quality settings for images that are to be read.

Refer to the following for details on this API

bColorDepth	bExOption	Scan result
EPS_BI_SCN_1BIT	EPS_BI_SCN_MANUAL	A value set to bThreshold is applied
	EPS_BI_SCN_SHARP	bThreshold is automatically set and the set value is applied
EPS_BI_SCN_8BIT	EPS_BI_SCN_MANUAL	No special handling is applied.
	EPS_BI_SCN_SHARP	Sharpening

Note

- ❏ This method does not affect previously read images.
- ❏ If bColorDepth is set to EPS_BI_SCN_8BITS(8), the parameter bThreshold is ignored.
- ❏ SCNSelectScanFace and SCNSelectScanUnit affect the scope of this method.

SCNSetImageFormat / SCNGetImageFormat

Set and get the format of image data returned by the driver.

Syntax

ErrorCode **SCNSetImageFormat**(*Format format*)

ErrorCode **SCNGetImageFormat**(*out Format format*)

Argument

Format:

format	Description
EPS_BI_SCN_TIFF	Data read as a TIFF file with CCITT (Group 4) compressed data
EPS_BI_SCN_TIFF256	Data read as a TIFF file with uncompressed 8bpp data
EPS_BI_SCN_JTIFF	Data read as a TIFF file with JPEG compressed data
EPS_BI_SCN_RASTER	Data read as raw raster data
EPS_BI_SCN_BITMAP	Data read with the windows bitmap format
EPS_BI_SCN_JPEGHIGH	Data read as a JPEG file with high compression
EPS_BI_SCN_JPEGNORMAL	Data read as a JPEG file with normal compression
EPS_BI_SCN_JPEGLOW	Data read as a JPEG file with low compression

Note

To save as an image file of ANSI X9.100-181-2007 standard, it must be a TIFF format with CCITT(Group 4) compression data (Format = EPS_BI_SCN_TIFF), black and white (bColorDepth of SCNSetImageQuality = EPS_BI_SCN_1BIT), and whose resolution is 200 dpi (Resolution of MFScan Class = MF_SCAN_DPI_200).

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

This method configures the data format used when transferring images back to the user. This is used as the base format for the generation of the Image object returned by the MFScan class Image property.

Note

This method does not affect previously read images.
If image codecs are not installed for all image formats it is possible that the MFScan.Image property will not be property set. In this case MFScan.Data can be used to access the data returned by the device.
SCNSelectScanFace and SCNSelectScanUnit affect the scope of this method.

SCNSetScanArea / SCNGetScanArea

Configure and read areas of check to scan.

Syntax

ErrorCode **SCNSetScanArea**(*System.Drawing.Rectangle area*)

ErrorCode **SCNSetScanArea**(*int startX, int startY, int endX, int endY*)

ErrorCode **SCNGetScanArea**(*out Rectangle area*)

ErrorCode **SCNGetScanArea**(*out int startX, out int startY, out int endX, out int endY*)

Argument

area: System.Drawing.Rectangle object showing scan area.

int startX: Starting horizontal coordinate of scan area.

int startY: Starting vertical coordinate of scan area.

int endX: Ending horizontal coordinate of scan area.

int endY: Ending vertical coordinate of scan area.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

Parameters are mapped as follows: area.Left = startX, area.Top = startY, area.Right = endX, area.Bottom = endY
This method does not affect previously read images.
Values supplied are in mm.
SCNSelectScanFace and SCNSelectScanUnit affect the scope of this method.

SCNSetCroppingArea

Set image cropping area.

Syntax

ErrorCode **SCNSetCroppingArea**(*byte areaNo, int startX, int startY, int endX, int endY*)

ErrorCode S**CNSetCroppingArea**(*byte areaNo, Rectangle area*)

Argument

areaNo: Identifier of the cropping area.

area: System.Drawing.Rectangle object showing scan area.

int startX: Starting horizontal coordinate of scan area.

int startY: Starting vertical coordinate of scan area.

int endX: Ending horizontal coordinate of scan area.

int endY: Ending vertical coordinate of scan area.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Note

Parameters are mapped as follows: area.Left = startX, area.Top = startY, area.Right = endX, area.Bottom = endY
This method does not affect previously read images.
Values supplied are in mm.
SCNSelectScanFace affect the scope of this method.

SCNGetCroppingAreas

Send and read raw data to and from the device.

Syntax

ErrorCode **SCNGetCroppingAreas**(*out byte[] areas*)

ErrorCode **SCNGetCroppingAreas**(*out Hashtable areas*)

Argument

areas (byte[]): Array of bytes with 5 bytes allocated for each cropping area, the first byte is the cropping area index, and the next 4 bytes are the startX, startY, endX and endY values as defined in the SCNSetCroppingArea method.

areas (Hashtable): Hashtable of Rectangle objects indexed with the cropping area identifier.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_BUFFER_OVER_FLOW	Read buffer over flow
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Note

SCNSelectScanFace affect the scope of this method.

SCNDeleteCroppingArea

Delete image cropping area.

Syntax

ErrorCode **SCNDeleteCroppingArea**(*byte areaNo*)

Argument

areaNo: Identifier of the cropping area.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Note

SCNSelectScanFace affect the scope of this method.

ESCNEnable

Determine where ESCN methods keep temporary data.

Syntax

static ErrorCode **ESCNEnable**(*Storage storage*)

Argument

Storage:

storage	Description
CROP_STORE_MEMORY	Store data in memory
CROP_STORE_FILE	Store data in file system

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_PARAM	Parameter error
ERR_ENABLE	OpenMonPrinter is already called

Note

This is a static method. It can and should be called before executing OpenMonPrinter.
Since this method is static, even if an error occurs it does not update the value of the LastError property.

ESCNSet AutoSize / ESCNGetAutoSize

Send and read if images should be sized automatically.

Syntax

ErrorCode **ESCNSetAutoSize**(*AutoSize autoSize*)

ErrorCode **ESCNGetAutoSize**(*out AutoSize autoSize*)

Argument

AutoSize:

autoSize	Description
CROP_AUTOSIZE_ENABLE	Automatically sizing enabled
CROP_AUTOSIZE_DISABLE	Automatically sizing disabled

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

ESCNSetCutSize / ESCNGetCutSize

Configure/Read image cut size.

Syntax

ErrorCode **ESCNSetCutSize**(*int cutSize*)

ErrorCode **ESCNGetCutSize**(*out int cutSize*)

Argument

cutSize Size to cut from edges in tenths of a millimeter.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

ESCNSet Rotate / ESCNGetRotate

Configure/Read image rotation setting.

Syntax

ErrorCode **ESCNSetRotate**(*Rotate rotate*)

ErrorCode **ESCNGetRotate**(*out Rotate rotate*)

Argument

Rotate:

rotate	Description
CROP_ROTATE_ENABLE	Image will be rotated 90 degrees
CROP_ROTATE_DISABLE	Image will not be rotated

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

ESCNSetDocumentSize / ESCNGetDocumentSize

Configure/Read size of document to scan.

Syntax

ErrorCode **ESCNSetDocumentSize**(*int documentWidth, int documentHeight*)

ErrorCode **ESCNGetDocumentSize**(*out int documentWidth, out int documentHeight*)

Argument

documentWidth: Width to scan in tenths of a millimeter.

documentHeight: Height to scan in tenths of a millimeter.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

ESCNSetDeSkew / ESCNGetDeSkew

Configure/ Read skew angle.

Syntax

ErrorCode **ESCNSetDeSkew**(*ushort wAngle*)

ErrorCode **ESCNGetDeSkew**(*out ushort lpwAngle*)

Argument

wAngle: Set or acquire a skew angle threshold value. (Unit: 0.01 degree)

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Note

Even if DESKEW_All is specified, Deskew is not activated unless the following conditions are met.

- The skew angle must be 10 degrees or less.
- The entire check image must be included within the range where the image can be scanned.

ESCNDefineCropArea

Set image cropping area.

Syntax

ErrorCode **ESCNDefineCropArea**(*byte areaNo, Rectangle area*)

ErrorCode **ESCNDefineCropArea**(*byte areaNo, int startX, int startY, int endX, int endY*)

Argument

areaNo: Identifier of the cropping area.

area: System.Drawing.Rectangle object showing scan area.

startX: Starting horizontal coordinate of scan area.

startY: Starting vertical coordinate of scan area.

endX: Ending horizontal coordinate of scan area.

endY: Ending vertical coordinate of scan area.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Note

Parameters are mapped as follows: area.Left = startX, area.Top = startY, area.Right = endX, area.Bottom = endY
Sizes are measured in tenths of a millimeter.

ESCNGetMaxCropAreas

Read maximum number of crop areas that can be defined.

Syntax

ErrorCode **ESCNGetMaxCropAreas**(*out int count*)

Argument

count: (out) Number of crop areas that can be defined.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

ESCNStoreImage

Send and read raw data to and from the device.

Syntax

ErrorCode **ESCNStoreImage**(*int fileIndex, String fileID, String imageTagData, byte cropAreaID*)

Argument

fileIndex: Index to identify data of the Crop image to be saved.

fileID: Identifier for the Crop image to be saved.

imageTagData: Tag data for the Crop image to be saved.

cropAreaID: ID of crop area to store.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_DISK_FULL	Capacity of a disk is insufficient
ERR_NO_IMAGE	No image data
ERR_ENTRY_OVER	Registration number of cases over
ERR_CROPAREAID	No specific CropAreaID
ERR_EXIST	Already exists
ERR_IMAGE_FILEOPEN	Open failure
ERR_IMAGE_FAILED	Image creation failure
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

Crop the CropArea from the image data saved in the WORK AREA by using cropAreaID selected and register the data to the Crop image saving table.

The cropped data are the same format as the original image data; however, if they are JPEG format, they are compressed and saved with JPEG standard compression.

If CropArea exceeds the range of image data of the WORK AREA, the excess area is solid white.

The cropped image data are saved in memory or a file by using the saving method selected with ESCNEnable.

When saving image data in a file, a file name is created by a device handle and identification data and is saved in the Windows temporary folder.

Note

String parameters cannot include the symbols \ / : , ; * ? " < > and |

ESCNRetreiveImage

Send and read raw data to and from the device.

Syntax

ErrorCode **ESCNRetrieveImage**(*int fileIndex, String fileID, String imageTagData, out byte[] data*)

ErrorCode **ESCNRetrieveImage**(*int fileIndex, String fileID, String imageTagData, out Image image*)

Argument

fileIndex: Index to identify data of the Crop image to be read.

fileID: Identifier for the Crop image to be read.

imageTagData: Tag data for the Crop image to be read.

data: (out) Image data as a byte array.

image: (out)Image data as an Image object.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_BUFFER_OVER_FLOW	Read buffer over flow
ERR_NOT_FOUND	Not found
ERR_IMAGE_FILEOPEN	Open failure
ERR_IMAGE_FILEREAD	Image file read error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

The data in writeCmd is sent to the device and its response is waited for timeout milliseconds and placed within the readBuff or response parameters.

Note

If NULL is selected for all identification data, the error (ERR_PARAM) is returned.
If Crop image data that match identification data selected are not present, the error (ERR_NOT_FOUND) is returned.
If the size of Crop image data is bigger than the memory size selected with pImageSize, the error (ERR_BUFFER_OVER_FLOW) is returned.
If Crop image data that match the selected identification data are present more than once, only the data that has been searched first is acquired.

ESCNClearImage

Send and read raw data to and from the device.

Syntax

ErrorCode **ESCNClearImage**(*ClearImageFlag flag, int fileIndex, String fileID, String imageTagData*)

Argument

flag: This specifies the deletion method. By using the deletion methods appropriately, it is possible to specify the Crop image data to be deleted. Refer to the explanation.

flag	Description
CROP_CLEAR_ALL_IMAGE	Deletes all the Crop image save data
CROP_CLEAR_BY_FILEINDEX	Deletes the Crop image data specified by dwFileIndex
CROP_CLEAR_BY_FILEID	Deletes the Crop image data specified by pFileID
CROP_CLEAR_BY_IMAGETAGDATA	Deletes the Crop image data specified by pImageTagData

fileIndex: Index to identify data of the Crop image to be cleared.

fileID: Identifier for the Crop image to be cleared.

imageTagData: Tag data for the Crop image to be cleared.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_NOT_FOUND	Not found
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

By using *Flag* appropriately, it is possible to specify the Crop image data to be deleted.

Example :

When the Crop image data specified with dwFileIndex and pImageTagData is to be deleted.

bFlag = CROP_CLEAR_BY_FILEINDEX + CROP_CLEAR_BY_IMAGETAGDATA;

ESCNGetRemainingImages

Acquire the remaining number of data of the Crop image that can be registered.

Syntax

ErrorCode **ESCNGetRemainingImages**(*out int count*)

Argument

count: The number of images that can be registered.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_NOT_FOUND	Not found
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

SCNMICRFunction

Scan ID card image.

Syntax

ErrorCode **SCNMICRFunction**(*FunctionType functionType*)

ErrorCode **SCNMICRFunction**(*MF mf, FunctionType functionType*)

Argument

FunctionType:

functionType	Description
MF_EXEC	Execute scanning
MF_CONTINUE	Continue scanning
MF_MICR_RETRANS	Resend MICR information
MF_SCAN_FRONT_RETRANS	Resend front face scan information
MF_SCAN_BACK_RETRANS	Resend back face scan information
MF_SET_BASE_PARAM	Set MFBase parameters
MF_SET_MICR_PARAM	Set MFMicr parameters
MF_SET_SCAN_PARAM	Set MFScan parameters for front face
MF_SET_SCAN_FRONT_PARAM	Set MFScan parameters for front face
MF_SET_SCAN_BACK_PARAM	Set MFScan parameters for back face
MF_SET_MICR_PARAM	Set MFMicr parameters
MF_SET_IQA_PARAM	Set MFIqa parameters
MF_SET_BARCODE_PARAM	Set MFBarcode parameters for front face
MF_SET_BARCODE_FRONT_PARAM	Set MFBarcode parameters for front face
MF_SET_BARCODE_BACK_PARAM	Set MFBarcode parameters for back face
MF_SET_PRINT_PARAM	Set MFPrint parameters
MF_SET_PROCESS_PARAM	Set MFProcess parameters
MF_CLEAR_BASE_PARAM	Clear MFBase parameters
MF_CLEAR_MICR_PARAM	Clear MFMicr parameters
MF_CLEAR_SCAN_PARAM	Clear MFScan parameters for front face
MF_CLEAR_SCAN_FRONT_PARAM	Clear MFScan parameters for front face
MF_CLEAR_SCAN_BACK_PARAM	Clear MFScan parameters for back face
MF_CLEAR_IQA_PARAM	Clear MFIqa parameters
MF_CLEAR_BARCODE_PARAM	Clear MFBarcode parameters for front face
MF_CLEAR_BARCODE_FRONT_PARAM	Clear MFBarcode parameters for front face
MF_CLEAR_BARCODE_BACK_PARAM	Clear MFBarcode parameters for back face
MF_CLEAR_PRINT_PARAM	Clear MFPrint parameters
MF_CLEAR_PROCESS_PARAM	Clear MFProcess parameters

functionType	Description
MF_GET_BASE_DEFAULT	Get MFBase parameters
MF_GET_MICR_DEFAULT	Get MFMicr parameters
MF_GET_SCAN_DEFAULT	Get MFScan parameters for front face
MF_GET_SCAN_FRONT_DEFAULT	Get MFScan parameters for front face
MF_GET_SCAN_BACK_DEFAULT	Get MFScan parameters for back face
MF_GET_IQA_DEFAULT	Get MFIqa parameters
MF_GET_BARCODE_DEFAULT	Get MFBarcode parameters for front face
MF_GET_BARCODE_FRONT_DEFAULT	Get MFBarcode parameters for front face
MF_GET_BARCODE_BACK_DEFAULT	Get MFBarcode parameters for back face
MF_GET_PRINT_DEFAULT	Get MFPrint parameters
MF_GET_PROCESS_DEFAULT	Get MFProcess parameters

mf: Reference to a MFBase, MFMicr, MFScan, MFIqa, MFBarcode, MFPrint or MFProcess object.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_OFFLINE	Device offline
ERR_PAPERINSERT_TIMEOUT	Image paper insert error of timeout
ERR_EXEC_FUNCTION	Other API is running
ERR_ABORT	Processing was canceled or near full is detected
ERR_MICR	Error occurred during MICR processing
ERR_SCAN	Error occurred during the image scan
ERR_PAPER_JAM	Paper jam has occurred
ERR_PAPER_INSERT	Paper insertion error
ERR_SCN_IQA	Error is detected by the IQA validation
ERR_BARCODE_NODA	Barcode cannot be detected.

Note

Refer to definition of MFBase, MFMicr, MFScan, MFIqa, MFBarcode, MFPrint and MFProcess for more information.

SCNMICRFunctionContinuously

Scan multiple checks.

Syntax

ErrorCode **SCNMICRFunctionContinuously**(*FunctionType functionType*)

ErrorCode **SCNMICRFunctionContinuously**(*MF mf, FunctionType functionType*)

Argument

FunctionType:

functionType	Description
MF_EXEC	Execute scanning
MF_CONTINUE	Continue scanning
MF_MICR_RETRANS	Resend MICR information
MF_SCAN_FRONT_RETRANS	Resend front face scan information
MF_SCAN_BACK_RETRANS	Resend back face scan information
MF_SET_BASE_PARAM	Set MFBase parameters
MF_SET_MICR_PARAM	Set MFMicr parameters
MF_SET_SCAN_PARAM	Set MFScan parameters for front face
MF_SET_SCAN_FRONT_PARAM	Set MFScan parameters for front face
MF_SET_SCAN_BACK_PARAM	Set MFScan parameters for back face
MF_SET_MICR_PARAM	Set MFMicr parameters
MF_SET_IQA_PARAM	Set MFIqa parameters
MF_SET_BARCODE_PARAM	Set MFBarcode parameters for front face
MF_SET_BARCODE_FRONT_PARAM	Set MFBarcode parameters for front face
MF_SET_BARCODE_BACK_PARAM	Set MFBarcode parameters for back face
MF_SET_PRINT_PARAM	Set MFPrint parameters
MF_SET_PROCESS_PARAM	Set MFProcess parameters
MF_CLEAR_BASE_PARAM	Clear MFBase parameters
MF_CLEAR_MICR_PARAM	Clear MFMicr parameters
MF_CLEAR_SCAN_PARAM	Clear MFScan parameters for front face
MF_CLEAR_SCAN_FRONT_PARAM	Clear MFScan parameters for front face
MF_CLEAR_SCAN_BACK_PARAM	Clear MFScan parameters for back face
MF_CLEAR_IQA_PARAM	Clear MFIqa parameters
MF_CLEAR_BARCODE_PARAM	Clear MFBarcode parameters for front face
MF_CLEAR_BARCODE_FRONT_PARAM	Clear MFBarcode parameters for front face
MF_CLEAR_BARCODE_BACK_PARAM	Clear MFBarcode parameters for back face
MF_CLEAR_PRINT_PARAM	Clear MFPrint parameters
MF_CLEAR_PROCESS_PARAM	Clear MFProcess parameters
MF_GET_BASE_DEFAULT	Get MFBase parameters
MF_GET_MICR_DEFAULT	Get MFMicr parameters

functionType	Description
MF_GET_SCAN_DEFAULT	Get MFScan parameters for front face
MF_GET_SCAN_FRONT_DEFAULT	Get MFScan parameters for front face
MF_GET_SCAN_BACK_DEFAULT	Get MFScan parameters for back face
MF_GET_IQA_DEFAULT	Get MFIqa parameters
MF_GET_BARCODE_DEFAULT	Get MFBarcode parameters for front face
MF_GET_BARCODE_FRONT_DEFAULT	Get MFBarcode parameters for front face
MF_GET_BARCODE_BACK_DEFAULT	Get MFBarcode parameters for back face
MF_GET_PRINT_DEFAULT	Get MFPrint parameters
MF_GET_PROCESS_DEFAULT	Get MFProcess parameters

mf: Reference to a MFBase, MFMicr, MFScan, MFIqa, MFBarcode, MFPrint or MFProcess object.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_OFFLINE	Device offline
ERR_WITHOUT_CB	Callback not registered
ERR_PAPERINSERT_TIMEOUT	Image paper insert error of timeout
ERR_EXEC_FUNCTION	Other API is running
ERR_ABORT	Processing was canceled or near full is detected
ERR_MICR	Error occurred during MICR processing
ERR_SCAN	Error occurred during the image scan
ERR_LINE_OVERFLOW	A line overflow has occurred during transaction printing
ERR_PAPER_PILED	Paper pilling error
ERR_PAPER_JAM	Paper jam has occurred
ERR_COVER_OPEN	Cover open error
ERR_MICR_NODATA	MICR data does not exist
ERR_MICR_BADDATA	MICR data is not able to recognize
ERR_MICR_NOISE	Noise error has occurred during MICR reading
ERR_SCN_COMPRESS	Scan image data compressing error
ERR_PAPER_EXIST	Because there is a paper on the path, API can not be execute

ErrorCode	Description
ERR_PAPER_INSERT	Paper insertion error
ERR_SCN_IQA	Error is detected by the IQA validation
ERR_BARCODE_NODA	Barcode cannot be detected.

Note

Refer to definition of MFBase, MFMicr, MFScan, MFIqa, MFBarcode, MFPrint and MFProcess for more information.

SCNMICRFunctionPostPrint

Scan check with endorsement determined after scanning.

Syntax

ErrorCode **SCNMICRFunctionPostPrint**(*FunctionType functionType*)

ErrorCode **SCNMICRFunctionPostPrint** (*MF mf, FunctionType functionType*)

Argument

FunctionType: Refer to "SCNMICRFunctionContinuously" on page 4-57.

mf: Reference to a MFBase, MFMicr, MFScan, MFIqa, MFBarcode, MFPrint or MFProcess object.

Return

Refer to "SCNMICRFunctionContinuously" on page 4-57.

Note

Refer to "SCNMICRFunctionContinuously" on page 4-57, MFBase, MFMicr, MFScan, MFIqa, MFBarcode, MFPrint and MFProcess for more information.

SCNMICRCancelFunction

Cancel operation of SCNMICRFunction, SCNMICRFunctionContinuously or SCNMICRFunctionPostPrint

Syntax

ErrorCode **SCNMICRCancelFunction**(*MfEjectType ejectType*)

Argument

MfEjectType: This specifies the destination of checks.

ejectType	Description
MF_EJECT_DISCHARGE	Eject checks into main pocket
MF_EJECT_RELEASE	A TM-J9000/J9100-compatible parameter (Eject checks into main pocket)

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_OFFLINE	Device offline
ERR_WITHOUT_CB	Callback not registered
ERR_EXEC_FUNCTION	Other API is running

GetMicrText

Read MICR data from scanned check.

Syntax

ErrorCode **GetMicrText**(*int transactionNumber, MFMicr mfMicr*)

Argument

transactionNumber: Transaction number of the check to read MICR information.

mfMicr: MFMicr object to receive MICR or OCR data.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_BUFFER_OVER_FLOW	Read buffer over flow
ERR_NOT_FOUND	Not found
ERR_MICR	Error occurred during MICR processing
ERR_NOT_EXEC	Processing is not executed
ERR_MICR_NODATA	MICR data is not existing
ERR_MICR_BADDATA	MICR data is not able to recognize
ERR_MICR_PARSE	MICR data can not be parsed
ERR_MICR_NOISE	Noise error has occurred during MICR reading

Explanation

Gets the MICR text read with SCNMICRFunctionContinuously / SCNMICRFunctionPostPrint. After the reading status MF_DATARECEIVE_DONE notification, the reading results are saved in the various parameters of the MFMicr object by specifying the relevant transaction number (ID) in transactionNumber.
When getting the MICR reading result, specify MF_MICR_USE_MICR in MicrOcrSelect of the MFMicr object, when getting the OCR reading result, specify MF_MICR_USE_OCR, and when getting the logical multiplication of the MICR and OCR reading results, specify MF_MICR_USE_MICR | MF_MICR_USE_OCR.
The MICR reading result and the OCR reading result are stored in MicrStr of the MFMicr object.
The reliability information of read text is stored in OcrReliableInfo of the MFMicr object.
When the MF_MICR_USE_MICR | MF_MICR_USE_OCR is specified in MicrOcrSelect, compares the MICR reading result with the OCR reading result. Different text is replaced by '?', and stored in MicrStr of the MFMicr object.

Note

- ❏ The interval during which the MICR reading result and OCR reading result corresponding to the transaction number (ID) can be acquired is from MF_DATARECEIVE_DONE notification to MF_CHECKPAPER_PROCESS_DONE notification.
- ❏ When the process is returned from the MF_CHECKPAPER_PROCESS_DONE notification handler to the API, the MICR reading result/OCR reading result saved by the API is discarded.
- ❏ When the font to read MICR (MFMicr.Font) is CMC7, the OCR recognition result (MFOcrAb.OcrReliableInfo) cannot be obtained. The relation between MicrOcrSelect and Font in MFMicr Class for OCR reading process is described below.

Font	**MicrOcrSelect**		
	MF_MICR_USE_MICR	**MF_MICR_USE_OCR**	**MF_MICR_USE_MICR | MF_MICR_USE_OCR**
MF_MICR_FONT_E13B	MICR magnetic waveform + OCR recognition	OCR recognition	MICR magnetic waveform + OCR recognition
MF_MICR_FONT_CMC7	MICR magnetic waveform	Error (ERR_MICR_NODATA)	Error (ERR_MICR_NODATA)

- ❏ Parsing can be used when the font to read MICR (MFMicr.Font) is E13B. When CMC7 is used, after calling GetMicrText, ERR_MICR_PARSE is returned as a return value.

MICRClearSpaces

Clears spaces included in MICR data obtained using GetMicrText.

Syntax

ErrorCode **MICRClearSpaces**(*RemoveSpace removeSpace*)

Argument

RemoveSpace: Specifies clearance of spaces in MICR data. The default value is CLEAR_SPACE_DISABLE.

removeSpace	Description
CLEAR_SPACE_DISABLE	Does not clear spaces in MICR data.
CLEAR_SPACE_ENABLE	Clears spaces in MICR data.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running

Explanation

Spaces in the following data in MFMicr Class are cleared by the API.

- MicrStr: MICR character string
- OcrReliableInfo: Information on reliability of MICR character recognition.

This API setting is valid until CloseMonPrinter is called.

SetOcrABAreaOrigin

Specifies the origin of the OCR area for MFOcrAb Class.

Syntax

ErrorCode **SetOcrABAreaOrigin**(*OcrOrigin eOcrOrigin*)

Argument

OcrOrigin: Specifies the origin of the OCR area. The default value is OCR_ORIGIN_TOP_LEFT.

eOcrOrigin	Description
OCR_ORIGIN_TOP_LEFT	Sets the origin to the top left corner of the document.
OCR_ORIGIN_BOTTOM_LEFT	Sets the origin to the bottom left corner of the document.
OCR_ORIGIN_TOP_RIGHT	Sets the origin to the top right corner of the document.
OCR_ORIGIN_BOTTOM_RIGHT	Sets the origin to the bottom right corner of the document.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running

Explanation

This API setting is valid until CloseMonPrinter is called.

GetOcrABText

Performs the OCR recognition for the OCR-A font or OCR-B font, and acquires the result.

Syntax

ErrorCode **GetOcrABText**(*uint dwTransactionNumber, OcrImageSource eImageSource, String szFileName, MFocrAB mfOcrAB*)

Argument

dwTransactionNumber: Transaction number of the check to read OCR information.

eImageSource: Specify an image targeted for the OCR recognition.

szFileName: Specify an image file name targeted for the OCR recognition.

mfOcrAB: MFOcrAB object to receive OCR data.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_HANDLE	Not opened correctly
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_NOT_FOUND	Not found
ERR_NOT_EXEC	Processing is not executed

Explanation

Necessary conditions for the OCR recognition other than the targeted images are set to the MF_OCR_AB structure.
The OCR recognition results are stored in the OUT attribute parameter of the MF_OCR_AB structure.
When OCR_SOURCE_TRANSACTION_NUMBER is specified to bImageSource, the target image for the OCR recognition becomes an image read immediately before the recognition and stored in the driver.
In this case, szFileName is ignored. After MF_DATARECEIVE_DONE is issued, set the corresponding transaction number to dwTransactionNumber in order to store the recognition result in the OUT attribute parameter of the MF_OCR_AB structure.
When OCR_SOURCE_IMAGE_FILE is specified to bImageSource, an image file stored by the driver is targeted for the OCR recognition. In this case, dwTransactionNumber is ignored.

This API can be executed anytime as long as an image file exists and satisfies the following conditions.

- Saved using SCNSetImageFormat specifying EPS_BI_SCN_BITMAP or EPS_BI_SCN_TIFF256.
- Saved using SCNSetImageQuality specifying EPS_BI_SCN_8BIT to the bColorDepth parameter.
- Saved using SCNSetImageQuality specifying EPS_BI_SCN_MANUAL to the bExOption parameter.

Note

When OCR_SOURCE_TRANSACTION_NUMBER is specified to bImageSource, acquiring the OCR recognition result corresponding to the specified transaction number (ID) is available from when MF_DATARECEIVE_DONE is issued until MF_CHECKPAPER_PROCESS_DONE is issued. When the MF_CHECKPAPER_PROCESS_DONE is sent from the notification handler to the StatusAPI, the OCR recognition result stored by the StatusAPI is discarded.

GetScanImage

Read image data from scanned check.

Syntax

ErrorCode **GetScanImage**(*int transactionNumber, MFScan mfScan*)

Argument

transactionNumber: Transaction number of the check to read image information.

mfScan: MFScan object to receive image data.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_NOT_FOUND	Not found
ERR_SCAN	Error occurred during the image scan
ERR_NOT_EXEC	Processing is not executed
ERR_SCN_COMPRESS	Scan image data compressing error

Explanation

Gets the scan image scanned with SCNMICRFunctionContinuously / SCNMICRFunctionPostPrint.
After the scanning status MF_DATARECEIVE_DONE notification, the scanning results are saved in the various parameters of the MFScan object by specifying the relevant transaction number (ID) in transactionNumber.
To get the front side scanning results, specify MF_SCAN_FACE_FRONT with SCNSelectScanFace, then execute this API.
To get the back side scanning results, specify MF_SCAN_FACE_BACK with SCNSelectScanFace, then execute this API.

Note

The SCNSetImageQuality, SCNSetImageFormat, and SCNScanArea setting values are reflected in the scanning result when SCNMICRFunctionContinuously / SCNMICRFunctionPostPrint is executed.
Set SCNImageQuality, SCNSetImageFormat, and SCNSetScanArea before executing SCNMICRFunctionContinuously / SCNMICRFunctionPostPrint scanning.
The interval during which the scan image corresponding to the transaction number (ID) can be acquired is from MF_DATARECEIVE_DONE notification to MF_CHECKPAPER_PROCESS_DONE notification.
When the process is returned from the MF_CHECKPAPER_PROCESS_DONE notification handler to the API, the scan image saved by the API is discarded.

GetBarcodeData

Decode a barcode from the scanned image data, and obtain the result.

Syntax

ErrorCode **GetBarcodeData** (*int transactionNumber, MFBarcode mfBarcode*)

Argument

transactionNumber: Specifies a transaction number for the BARCODE decode.

mfBarcode: MFBarcode class object to store the result of decode.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Success
ERR_NO_MEMORY	Memory is insufficient
ERR_HANDLE	The handle value that specifies the device is incorrect
ERR_PARAM	Parameter error
ERR_NOT_FOUND	No data error
ERR_EXEC_FUNCTION	Cannot be used because the other API is being executed
ERR_NOT_EXEC	Processing is not executed
ERR_BARCODE_NODATA	Barcode cannot be detected.

Explanation

Decodes a barcode from the scanned image data, and obtains the result.
The result of barcode decode is set in the propaty of OUT attribute in MFBarcode class. For the details, refer to "MFBarcode Class - Properties" on page 4-154.
To decode a barcode, specify the front/back side with SCNSelectScanFace.

Note

- ❏ The decoding result of BARCODE corresponding to the transaction number is obtainable from the processing status of MF_DATARECEIVE_DONE to the notification of MF_CHECKPAPER_PROCESS_DONE. The decoding result of barcode kept by Status API is discarded when the handler of MF_CHECKPAPER_PROCESS_DONE sends back the processing to Status API.
- ❏ When decoding multiple barcodes at the same time, the correct result cannot be obtained if the barcodes other than given below are decoded.
  - Barcode symbols are the same
  - Barcodes are lined in the horizontal direction
- ❏ This API is MF_SET_BARCODE_FRONT_PARAM or MF_SET_BARCODE_BACK_PARAM, and can obtain the decoding result different from the MFBarcode Class which was set earlier. In such a case, where to eject or the flanking process is set according to the current setting. The decoding result can be obtained even if the MFBarcode Class is not set earlier.

DecodeBarcode

Decode a barcode from the specified image file, and obtain the result.

Syntax

ErrorCode **DecodeBarcode** (*String szFileName, MFBarcode mfBarcode*)

Argument

szFileName: Name of the image file (file pass)

mfBarcode: MFBarcode class object to store the result of decode.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Success
ERR_NO_MEMORY	Memory is insufficient
ERR_HANDLE	The handle value that specifies the device is incorrect
ERR_PARAM	Parameter error
ERR_IMAGE_FILEOPEN	Open failure
ERR_IMAGE_UNKNOWNFORMAT	Format injustice
ERR_BARCODE_NODATA	Barcode cannot be detected.

Explanation

Decodes a barcode from the image file, and obtains the result.
The result of barcode decode is set in the propaty of OUT attribute in MFBarcode class. For the details, refer to "MFBarcode Class - Properties" on page 4-154.

Note

- ❏ Make sure to use the image file created in the following way.
  - Saved file with EPS_BI_SCN_BITMAP or EPS_BI_SCN_TIFF256 specified in SCNSetImageFormat
  - Saved file with EPS_BI_SCN_8BIT to bColorDepth specified in SCNSetImageQuality
  - Saved file with EPS_BI_SCN_SHARP to bExOption specified in SCNSetImageQuality
- ❏ When decoding multiple barcodes at the same time, the correct result cannot be obtained if the barcodes other than given below are decoded.
  - Barcode symbols are the same
  - Barcodes are lined in the horizontal direction

DecodeBarcodeMemory

Decode the barcode from the image data on the application memory, and obtain the result.

Syntax

ErrorCode **DecodeBarcodeMemory** (*Bitmap objBitmap, MFBarcode mfBarcode*)

Argument

objBitmap: Image object for decode

mfBarcode: MFBarcode class object to store the result of decode.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Success
ERR_NO_MEMORY	Memory is insufficient
ERR_HANDLE	The handle value that specifies the device is incorrect
ERR_PARAM	Parameter error
ERR_IMAGE_UNKNOWNFORMAT	Format injustice
ERR_BARCODE_NODATA	Barcode cannot be detected.

Explanation

The result of barcode decode is set in the propaty of OUT attribute in MFBarcode class. For the details, refer to "MFBarcode Class - Properties" on page 4-154.

Note

- ❏ Make sure to use the image file created in the following way.
  - Saved file with EPS_BI_SCN_BITMAP or EPS_BI_SCN_TIFF256 specified in SCNSetImageFormat
  - Saved file with EPS_BI_SCN_8BIT to bColorDepth specified in SCNSetImageQuality
  - Saved file with EPS_BI_SCN_SHARP to bExOption specified in SCNSetImageQuality
- ❏ When decoding multiple barcodes at the same time, the correct result cannot be obtained if the barcodes other than given below are decoded
  - Barcode symbols are the same
  - Barcodes are lined in the horizontal direction

UpdateEndorseText

Send and read raw data to and from the device.

Syntax

ErrorCode **UpdateEndorseText**(*String[] strings, int[] attributes, FontType[] fonts, FontScale[] fontSizes*)

ErrorCode **UpdateEndorseText**(*MFPrint data*)

Argument

strings: Array of strings for endorsement text.

attributes: Array of attribute values for endorsement text.

fonts: Array of fonts for endorsement text.

fontSizes: Array of font size values for endorsement text.

data: MFPrint object String, Attribute, Font and FontSize properties are used to update the endorsement text.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_NOT_EXEC	Processing is not executed

Explanation

The endorsed text can be updated by invoking this API from the SCNMICRFunctionPostPrint scanning status MF_DATARECEIVE_DONE notification handler. If this API is executed when the MF_PRINT structure is not set in SCNMICRFunctionPostPrint without performing endorsed printing, the ERR_EXEC_FUNCTION error is returned and the endorsed text is not updated.

GetTransactionNumber / SetTransactionNumber

Read and configure the transaction number of the next scanned check.

Syntax

ErrorCode **GetTransactionNumber**(*out int transactionNumber*)

ErrorCode **SetTransactionNumber**(*int transactionNumber*)

Argument

transactionNumber: Transaction number of the next scanned check.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

SetTransactionNumberWithIncrement

Configure the transaction number and the rate at which it increases.

Syntax

ErrorCode **SetTransactionNumberWithIncrement**(*int number, int increment*)

Argument

number: Transaction number of the next scanned check.

increment: The rate at which the transaction number will increment.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

PrintText

Output text to the device to be printed.

Syntax

ErrorCode **PrintText**(*String text, MFDecorate decorate*)

Argument

text: Text to be output.

decorate: MFDeviceFont or MFTrueType object defining text format view description of MFDeviceFont and MFTrueType for more information.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error

PrintImage

Send bitmap image to the device to be printed.

Syntax

ErrorCode **PrintImage**(*String filename*)

Argument

filename: Name of bitmap file to print.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_IMAGE_FILEOPEN	Open failure
ERR_IMAGE_UNKNOWNFORMAT	Invalid format
ERR_SIZE	Size excess error

PrintMemoryImage

Send bitmap image to the device to be printed.

Syntax

ErrorCode **PrintMemoryImage**(*Bitmap objImage*)

Argument

objImage: Sets BitmapObject.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_IMAGE_UNKNOWNFORMAT	Invalid format
ERR_SIZE	Size excess error

SetPrintSize / GetPrintSize

Configure and Read the size of Barcode and Image to print.

Syntax

ErrorCode **SetPrintSize**(*int width, int height*)

ErrorCode **GetPrintSize**(*out int width, out int height*)

Argument

width: Width in millimeters.

height: Height in millimeters.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

SetPrintPosition / GetPrintPosition

Configure and Read the location of output for print commands.

Syntax

ErrorCode **SetPrintPosition**(*int horizontal, int vertical*)

ErrorCode **GetPrintPosition**(*out int horizontal, out int vertical*)

Argument

horizontal: Horizontal location in millimeters.

vertical: Vertical location in millimeters.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

SetEndorseDirection

Specifies the direction of E-Endorse.

Syntax

ErrorCode **SetEndorseDirection**(*ElectricEndorseDirection eDirection*)

Argument

ElectricEndorseDirection:

Specifies the direction for E-endorse. The default value is EENDORSE_DIRECTION_LEFTRIGHT.

eDirection	Description
EENDORSE_DIRECTION_LEFTRIGHT	From left to right (normal direction)
EENDORSE_DIRECTION_TOPBOTTOM	From top to bottom (Rotate 90° clockwise)
EENDORSE_DIRECTION_RIGHTLEFT	From right to left (upside down)
EENDORSE_DIRECTION_BOTTOMTOP	From bottom to top (Rotate 90° counterclockwise)

Return value

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error

Explanation

The electric endorse is rotated around the supporting point specified with SetPrintPosition

This API setting is applied when PrintText, PrintImage, or PrintMemoryImage is called.

SetPrintStation / GetPrintStation

Configure and Read where printing will occur.

Syntax

ErrorCode **SetPrintStation**(*PrintingStation station*)

ErrorCode **GetPrintStation**(*out PrintingStation station*)

Argument

PrintingStation:

station	Description
MF_ST_E_ENDORSEMENT	Endorsement on check print station
MF_ST_E_ENDORSEMENT_BACK	Same as MF_ST_E_ENDORSEMENT.
MF_ST_E_ENDORSEMENT_FRONT	Sets electronic endorsement to front side as the station for printing.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

BufferedPrint

Configure buffering of print jobs.

Syntax

ErrorCode **BufferedPrint**(*PrintBuffer function*)

Argument

PrintBuffer:

function	Description
MF_PRT_BUFFERING	Start buffering of print commands
MF_PRT_EXEC	Execute buffered print commands
MF_PRT_CLEAR	Delete buffered print commands

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

SetStatusBack

Enable the StatusCallback and StatusCallbackEx events.

Syntax

ErrorCode **SetStatusBack** ()

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

CancelStatusBack

Disable the StatusCallback and StatusCallbackEx events.

Syntax

ErrorCode **CancelStatusBack** ()

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly

SCNMICRSetStatusBack

Enable the SCNMICRStatusCallback event.

Syntax

ErrorCode **SCNMICRSetStatusBack** ()

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_EXEC_FUNCTION	Other API is running

SCNMICRCancelStatusBack

Disable the SCNMICRStatusCallback event.

Syntax

ErrorCode **SCNMICRCancelStatusBack** ()

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_EXEC_FUNCTION	Other API is running

SetNumberOfDocuments

Specifies the number of docments to read using SCNMICRFunctionContinuously.

Note

This API setting is available when the processing mode is set to High-speed mode. In Confirmation mode, this API setting is ignored.

Syntax

ErrorCode **SetNumberOfDocuments** (*byte bNumber*)

Argument

bNumber: Specifies the number of documents to read. Valid values are from 0 to 100. The default value is 0.

Return value

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_LESS_CHECKS	The number of checks specified for SetNumberOfDocuments cannot be read.

Explanation

When 0 is specified, the number of document to read is unlimited. When 1 to 100 is specified, after the specified number of documents is read, the process is stopped. This AIP setting is valid until CloseMonPrinter is called.
If reading of the documents set in the ASF is completed before reading the number of documents specified using this API, the error code (ERR_LESS_CHECKS) is sent, indicating that the specified number of documents have not been read to subStatus of the reading status MF_FUNCTION_DONE.

SetBehaviorToScnResult

Specify an operation in Confirmation mode.

Syntax

ErrorCode **SetBehaviorToScnResult** (*MfEject bEject, MfStamp bStamp, MfProcessContinue bNextCheck*)

Argument

MfEject:

bEject	Description
MF_EJECT_MAIN_POCKET	Eject paper to the main pocket
MF_EJECT_SUB_POCKET	Eject paper to the sub pocket
MF_EJECT_NOEJECT	Not eject paper

MfStamp:

bStamp	Description
MF_STAMP_ENABLE	Stamp paper
MF_STAMP_DISABLE	Not stamp paper

MfProcessContinue:

bNextCheck	Description
MF_PROCESS_CONTINUE_OVERLAP	Read the next check paper (overlap processing)
MF_PROCESS_CONTINUE_NOOVERLAP	Read the next check paper (No overlap processing)
MF_PROCESS_CONTINUE_CANCEL	Not read the next check paper

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error

Note

This function can be called from a callback function when ActivationMode of the MFProcess Class is set to MF_ACTIVATE_MODE_CONFIRMATION. When this function is called specifying MF_ACTIVATE_MODE_HIGH_SPEED, the function is ignored.

SetPaperThickness

Specifies the double feed threshold.

Syntax

ErrorCode **SetPaperThickness** (*ushort wThreshold*)

Argument

wThreshold: Specifies the double feed threshold. The valid specification range is 1 to 40 (0.01 mm to 0.40 mm).
When "0" is specified, the default value is selected.
Paper thickness exceeding the specified value is detected as double feed.
The default value is listed below.

MfPaperType	Default value
MF_PAPER_TYPE_CHECK	0.17 mm
MF_PAPER_TYPE_OTHER	0.23 mm

Return value

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running

Explanation

When the double feed threshold is specified by this API, the setting of bPaperType in MFProcess Class is ignored. This API setting is valid until CloseMonPrinter is called.

RingBuzzer

Configure the internal buzzer sound settings.

Syntax

ErrorCode **RingBuzzer** (*MfBuzzerTone bTone, byte bCount, ushort wOnTime, ushort wOffTime*)

Argument

MfBuzzerTone:

bTone	Description
MF_BUZZER_TONE_HIGH	Sound high-pitched buzzer sounds
MF_BUZZER_TONE_MIDDLE	Sound middle-pitched buzzer sounds
MF_BUZZER_TONE_LOW	Sound low-pitched buzzer sounds

bCount: Specify the number of buzzers.

wOnTime: Specify the buzzer tone duration in units of msec.

wOffTime: Specify the off time of buzzer tone in units of msec.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error

Note

Setting all the parameters; bTone, bCount, wOnTime, wOffTime, to 0 (zero) stops the buzzer. This also applies to the buzzer specified by the MFBase structure.

SetWaterfallMode

Enables/Disables the Waterfall mode.

Syntax

ErrorCode **SetWaterfallMode**(*WaterfallMode eWaterfallMode*)

Argument

WaterfallMode: Enables/Disables the Waterfall mode. The default value is WATERFALL_MODE_DISABLE.

eWaterfallMode	Description
WATERFALL_MODE_DISABLE	Disables Waterfall mode.
WATERFALL_MODE_STANDARD	Enables Waterfall mode. Ejects to the main pocket when starting the reading process. When the main pocket's near full is detected, the ejection pocket is switched to the sub pocket. When the sub pocket's near full is detected, the ejection pocket is switched to the main pocket.
WATERFALL_MODE_INHERIT_POCKET	Enables Waterfall mode. Ejects to the ejection pocket of the previous reading process. (For example, the previous ejection pocket is the sub pocket, ejects to the sub pocket.) When a pocket near full is detected, the ejection pocket is switched to the other pocket. When a pocket near full has already been detected when starting the reading process, the ejection pocket is switched to the other pocket. The ejection pocket, however, is not switched when the other pocket's near full has been detected.

Return value

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_EXEC_FUNCTION	Other API is running

Explanation

When a pocket near full is detected when the Waterfall mode is started, the first document is ejected to the following pocket.

Waterfall mode	Pocket status		Ejection pocket
	Main Pocket	Sub Pocket	
WATERFALL_MODE_STANDARD	Not NearFull	Not NearFull	Main Pocket
	NearFull	Not NearFull	Sub Pocket
	Not NearFull	NearFull	Main Pocket
	NearFull	NearFull	Main Pocket
WATERFALL_MODE_INHERIT_POCKET	Not NearFull	Not NearFull	Inherit Pocket
	NearFull	Not NearFull	Sub Pocket
	Not NearFull	NearFull	Main Pocket
	NearFull	NearFull	Inherit Pocket

Note

- ❏ When a pocket near full is detected from both pockets, it is recommended to set NearFullSelect of MFProcess Class to MF_NEARFULL_NOT_PERMIT, to stop the reading process. (See "NearFullSelect" on page 4-130.)
- ❏ When the Waterfall mode is enabled, the ejection pocket setting for MFProcess Class is ignored.
- ❏ When this API setting is executed in the Confirmation mode, the ejection pocket setting for SetBehaviorToScnResult has the priority.
- ❏ When this API is called when reading is being processed, the error code (ERR_EXEC_FUNCTION) is returned.
- ❏ This API setting is valid until CloseMonPrinter is called.

GetIqaResult

Gets the IQA validation result.

Syntax

ErrorCode **GetIqaResult**(*int dwTransactionNumber, MFIqaResult pResult*)

Argument

dwTransactionNumber: Specifies the transaction number of the target to get the IQA validation result.

pResult: Specifies the MFIqaResult class object to save the IQA validation result. The following result is saved for each IQA validation item.

Constant	Description
IQARESULT_NOT_TESTED	IQA validation is not executed.
IQARESULT_PASS	Passed IQA validation.
IQARESULT_NOT_PASS	Not passed IQA validation.

Return value

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_FOUND	Not found
ERR_NOT_EXEC	Processing is not executed

Explanation

For details of the IQA validation result, see "MFIqaResult Class - Properties" on page 4-147.

GetOfflineCodeByIndex

Acquires data that shows the cause of the device to go offline.

Syntax

ErrorCode **GetOfflineCodeByIndex** (*int iIndex, out byte lpbOfflinecode*)

Argument

iIndex: Specify which byte that shows one of offline causes to be acquired. A value of 1 to 5 can be used to specify the byte.

lpbOfflinecode: Specify memory address to which the acquired offline cause is stored.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_EXEC_FUNCTION	Other API is running
ERR_RESET	Now device is resetting

Explanation

The offline causes assigned to each byte are shown in the following tables, refer to "Offline Code (GetOfflineCodeByIndex)" on page 4-8.

Note

When this function is executed while the device is online, zero will be written to the address specified with lpbOfflinecode parameter.

GetVersion

Acquires the version information of the driver or the module used by the driver.

Syntax

ErrorCode **GetVersion** (*DriverType eDriverType, VersionType eType, MFVersion ptVersion*)

Argument

DriverType: The driver type of the version information to be acquired. One of the following values can be specified.

eDriverType	Description
DRIVER_TYPE_J9000	TM-J9000+ Driver
DRIVER_TYPE_S1000	TM-S1000 Driver

VersionType: The type of the version to be acquired. One of the following values can be specified.

eType	Description
VERSION_TYPE_DRIVER	Driver version
VERSION_TYPE_USB	TMUSBDriver version
VERSION_TYPE_MICR	Magnetic waveform analysis module version
VERSION_TYPE_MICR_E13B	Magnetic waveform analysis module version (E13B)
VERSION_TYPE_MICR_CMC7	Magnetic waveform analysis module version (CMC7)
VERSION_TYPE_OCR	OCR recognition module version
VERSION_TYPE_IMAGE	Image processing module version
VERSION_TYPE_IQA	IQA module version
VERSION_TYPE_BARCODE	Barcode module version

ptVersion: MFVersion object to receive the version information.

Return

One of the following values is returned.

ErrorCode	Description
SUCCESS	Method completed successfully
ERR_PARAM	Parameter error
ERR_NOT_FOUND	Not found

Properties

This sections contains descriptions of all of the properties exposed by the MFDevice class.

IsValid

Type

Readonly: *bool*

Values

bool	Description
true	instance has successfully connected to device
false	instance has not successfully connected to device

LastError

Type

Readonly: *ErrorCode*

Values

ErrorCode	Description
SUCCESS	Success
ERR_TYPE	Parameter error
ERR_OPENED	Device already opened
ERR_NO_PRINTER	Specified device does not exist
ERR_NO_TARGET	Invalid device specified
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_HANDLE	Not opened correctly
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_NOT_SUPPORT	Method not supported
ERR_OFFLINE	Device offline
ERR_NOT_EPSON	Connected device not an Epson device
ERR_WITHOUT_CB	Callback not registered
ERR_BUFFER_OVER_FLOW	Read buffer over flow
ERR_REGISTRY	Registry error
ERR_ENABLE	OpenMonPrinter is already called
ERR_DISK_FULL	Capacity of a disk is insufficient
ERR_NO_IMAGE	No image data
ERR_ENTRY_OVER	Registration number of cases over
ERR_CROPAREAID	No specific CropAreaID
ERR_EXIST	Already exists
ERR_NOT_FOUND	Not found

ErrorCode	Description
ERR_IMAGE_FILEOPEN	Open failure
ERR_IMAGE_UNKNOWNFORMAT	Invalid format
ERR_IMAGE_FAILED	Image creation failure
ERR_IMAGE_FILEREAD	Image file read error
ERR_PAPERINSERT_TIMEOUT	Image paper insertion timeout
ERR_EXEC_FUNCTION	Other API is running
ERR_EXEC_MICR	Now reading MICR
ERR_EXEC_SCAN	Now scanning image
ERR_RESET	Now device is resetting
ERR_THREAD	Failure to the start of thread
ERR_ABORT	Processing was canceled or near full is detected
ERR_MICR	Error occurred during MICR processing
ERR_SCAN	Error occurred during the image scan
ERR_LINE_OVERFLOW	A line overflow has occurred during transaction printing
ERR_NOT_EXEC	Processing is not executed
ERR_SIZE	Size excess error
ERR_PAPER_PILED	Paper pilling error
ERR_PAPER_JAM	Paper jam has occurred
ERR_COVER_OPEN	Cover open error
ERR_MICR_NODATA	Data does not exist
ERR_MICR_BADDATA	MICR data is not able to recognize
ERR_MICR_PARSE	Data can not be parsed
ERR_MICR_NOISE	Noise error has occurred during MICR reading
ERR_SCN_COMPRESS	Scan image data compressing error
ERR_PAPER_EXIST	Because there is a paper on the path, API can not be execute
ERR_PAPER_INSERT	Paper insertion error

Status

Type

Readonly: *ASB*

Values

Refer to "Device Status" on page 4-1.

OfflineCode

Type

Readonly: *byte[5]*

Values

Refer to "Offline Code (OfflineCode)" on page 4-6.

ESCNAutoSize

Type

Read/Write: *AutoSize*

Values

AutoSize	Description
CROP_AUTOSIZE_ENABLE	Automatically sizing enabled
CROP_AUTOSIZE_DISABLE	Automatically sizing disabled

ESCNCutSize

Type

Read/Write: *int*

Values

Size to cut from edges in tenths of a millimeter.

ESCNRotate

Type

Read/Write: *Rotate*

Values

Rotate	Description
CROP_ROTATE_ENABLE	Image will be rotated 90 degrees
CROP_ROTATE_DISABLE	Image will not be rotated

ESCNDocumentSize

Type

Read/Write: *System.Drawing.Size*

Values

width and height of document in tenths of a millimeter.

ESCNDeSkew

Type

Read/Write: *uShort*

Values

Specify or acquire a threshold value of the skew angle. (unit: 0.01 degree)

ESCNRemainingImages

Type

Readonly: *int*

Values

The remaining number of data of the Crop image that can be registered.

TransactionNumber

Type

Read/Write*:* *int*

Values

Identifying number of the next check to be scanned.

TransactionIncrement

Type

Writeonly*:* *int*

Values

Increment at which the transaction number increases.

PrintStation

Type

Read/Write*:* *PrintingStation*

Values

PrintingStation	Description
MF_ST_ROLLPAPER	Roll paper print station
MF_ST_VALIDATION	Validation on slip print station
MF_ST_E_ENDORSEMENT	Endorsement on check print station

PrintSize

Type

Read/Write*:* *System.Drawing.Size*

Values

Width and height of image or barcode in millimeters.

PrintPosition

Type

Read/Write*:* *System.Drawing.Point*

Values

Horizontal and Vertical location of next print command.

Events

This sections contains descriptions of all of the events exposed by the MFDevice class.

StatusCallback / StatusCallbackEx

Event when the status of the device changes.

Event

event StatusCallbackHandler StatusCallback;

event StatusCallbackHandlerEx StatusCallbackEx;

Delegate

delegate void StatusCallbackHandler(*ASB asb*);

delegate void StatusCallbackHandlerEx(*ASB asb, String portName*);

Argument

asb: Combination of device status bits. Refer to "Device Status" on page 4-1.

portName: Port to which the device is connected.

MfDone

Event when SCNMICRFunction operation is complete.

Event

event MfDoneHandler MfDone;

Delegate

delegate void MfDoneHandler();

SCNMICRStatusCallback

Event when the stage of SCNMICRFunctionContinuously or SCNMICRFunctionPostPrint operation changes.

Event

event SCNMICRStatusCallbackHandler SCNMICRStatusCallback;

Delegate

delegate void SCNMICRStatusCallbackHandler(*int TransactionNumber, MainStatus mainStatus, ErrorCode subStatus, String portName*);

Argument

TransactionNumber: Indentification number for this check in the transaction.

MainStatus: Combination of device status bits.

mainStatus	Description
MF_FUNCTION_START	Started check paper scanning
MF_CHECKPAPER_PROCESS_START	Inserted check paper and started scanning
MF_DATARECEIVE_START	Started receiving data
MF_DATARECEIVE_DONE	Completed receiving data
MF_CHECKPAPER_PROCESS_DONE	Ejected check paper and completed the process
MF_FUNCTION_DONE	Finished check paper scanning
MF_ERROR_OCCURED	Error occurred while scanning, view subStatus for details

ErrorCode: combination of device status bits.

subStatus	Description
ERR_PAPER_PILED	Detected double feed
ERR_PAPER_JAM	Paper jam has occurred
ERR_COVER_OPEN	Cover open error
ERR_MICR_NODATA	MICR data is not existing
ERR_MICR_BADDATA	MICR data is not able to recognize
ERR_MICR_NOISE	Noise error has ocurred during MICR reading

portName: Port to which the device is connected.

MFBase Class - Properties

This sections contains descriptions of all of the properties exposed by the MFBase class.

Version

Type

Readonly: *int*

Values

The version of structure used internally within this class.

Ret

Type

Readonly: *ErrorCode*

Values

ErrorCode	Description
SUCCESS	Success
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_PAPERINSERT_TIMEOUT	Image paper insert error of timeout
ERR_ABORT	Processing was canceled or near full is detected
ERR_MICR	Error occurred during MICR processing
ERR_SCAN	Error occurred during the image scan
ERR_NOT_EXEC	Processing is not executed

Timeout

Type

Read/Write*:* *int*

Values

The time to wait before paper insertion is specified in seconds. The values that can be specified are 0 to 300 (five minutes), and when 0 is specified, there is no timeout. When there is a timeout, ERR_PAPERINSERT_TIMEOUT is returned. Timeouts other than paper insertion cannot be specified.

ErrorEject

Type

Read/Write*:* *MfEjectType*

Values

MfEjectType	Description
MF_EJECT_DISCHARGE	Eject check to main pocket
MF_EJECT_RELEASE	Eject check to sub pocket

SuccessEject

Type

Read/Write*:* *MfEjectType*

Values

MfEjectType	Description
MF_EJECT_DISCHARGE	Eject check to main pocket
MF_EJECT_RELEASE	Eject check to sub pocket

BuzzerHz

Type

Read/Write*:* BuzzerHzClass class with indexer of type MfBuzzerType for type MfBuzzerHz

Indexer

Indexer	Description
MF_BUZZER_TYPE_SUCCESS	Successful read of check
MF_BUZZER_TYPE_ERROR	Error occurred while reading check
MF_BUZZER_TYPE_WFEED	Double check feed detected

Values

MfBuzzerHz	Description
MF_BUZZER_HZ_440	Set buzzer frequency to 440Hz
MF_BUZZER_HZ_880	Set buzzer frequency to 880Hz
MF_BUZZER_HZ_4000	Set buzzer frequency to 4000Hz

BuzzerCount

Type

Read/Write: BuzzerCountClass class with indexer of type MfBuzzerType for type MfBuzzerCount

Indexer

Indexer	Description
MF_BUZZER_TYPE_SUCCESS	Successful read of check
MF_BUZZER_TYPE_ERROR	Error occurred while reading check
MF_BUZZER_TYPE_WFEED	Double check feed detected

Values

MfBuzzerCount	Description
MF_BUZZER_DISABLE	Do not sound buzzer
MF_BUZZER_COUNT_ONE	Sound buzzer once
MF_BUZZER_COUNT_TWO	Sound buzzer twice
MF_BUZZER_COUNT_MAX	Sound buzzer three times

PortName

Type

Readonly: *String*

Values

The port name associated with this device.

MFMicr Class - Properties

This sections contains descriptions of all of the properties exposed by the MFMicr class.

Version

Type

Readonly: *int*

Values

The version of structure used internally within this class.

Ret

Type

Readonly: *ErrorCode*

Values

ErrorCode	Description
SUCCESS	Success
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_NOT_FOUND	Not found
ERR_ABORT	Processing was canceled or near full is detected
ERR_MICR	Error occurred during MICR processing
ERR_NOT_EXEC	Processing is not executed

Font

Type

Read/Write*:* *MfMicrFont*

Values

MfMicrFont	Description
MF_MICR_FONT_E13B	Read MICR characters with the E13B font
MF_MICR_FONT_CMC7	Read MICR characters with the CMC7 font (not supported)

MicrOcrSelect

Type

Read/Write: *MfMicrType*

Values

MfMicrType	Description
MF_MICR_USE_MICR	Read MICR characters magnetically
MF_MICR_USE_OCR	Read MICR characters optically
MF_MICR_USE_BOTH	Read MICR characters both magnetically and optically

Parsing

Type

Read/Write: *bool*

Values

bool	Description
true	Parse magnetically read characters
false	Do not parse magnetically read characters

Status

Type

Readonly: *byte*

Values

Bit values of the byte are as follows.

Bit	Function	Value	
		0	1
0	Reading font	E13B	CMC7
1,2	Reserved	Fixed to 0	
3	Detailed information	-	Added
4	Reread	-	Disabled
5	Reading results	Success	Failure
6	OCR processing error	No	Yes
7	Readout data reception error	No	Yes

Detail

Type

Readonly*:* *byte*

Values

Values of the byte are as follows.

Value	Information
40h	Success
41h	Check reading was not executed even once
44h	A check of irregular length was inserted
45h	The magnetic waveform was not detected
46h	Characters which could not be analyzed were detected in analysis processing
47h	An error occurred during the check reading
48h	An error was detected in the noise measurement
49h	Check reading was stopped due to a transport error
4Bh	A paper length error occurred during check reading

MicrStr

Type

Readonly: *String*

Values

Raw data read magnetically from MICR characters.

AccountNumber

Type

Readonly: *String*

Values

Account number parsed from data read magnetically from MICR characters.

Amount

Type

Readonly: *String*

Values

Amount parsed from data read magnetically from MICR characters.

BankNumber

Type

Readonly: *String*

Values

Bank number parsed from data read magnetically from MICR characters.

SerialNumber

Type

Readonly: *String*

Values

Serial number parsed from data read magnetically from MICR characters.

EPC

Type

Readonly: *String*

Values

EPC property parsed from data read magnetically from MICR characters.

TransitNumber

Type

Readonly: *String*

Values

Transit number parsed from data read magnetically from MICR characters.

CheckType

Type

Readonly: *Check*

Values

Check	Description
CHECK_PERSONAL	Check type is personal
CHECK_BUSINESS	Check type is business
CHECK_UNKNOWN	Check is of unknown type

CountryCode

Type

Readonly: *Country*

Values

Country	Description
COUNTRY_USA	Check is from the United States
COUNTRY_CANADA	Check is from Canada
COUNTRY_MEXICO	Check is from Mexico
COUNTRY_UNKNOWN	Country of origin could not be determined

OcrReliableInfo.FirstSelectString

Type

Readonly: *String*

Values

Best guess raw data read optically from MICR characters.

OcrReliableInfo.SecondSelectString

Type

Readonly: *String*

Values

Second guess raw data read optically from MICR characters.

OcrReliableInfo.FirstSelectPercentage

Type

Readonly: *int[]*

Values

Confidence level of best guess raw data read optically from MICR characters.

OcrReliableInfo.SecondSelectPercentage

Type

Readonly: *int[]*

Values

Confidence level of second guess raw data read optically from MICR characters.

MFScan Class - Properties

This sections contains descriptions of all of the properties exposed by the MFScan class.

Version

Type

Readonly: *int*

Values

The version of structure used internally within this class.

Ret

Type

Readonly: *ErrorCode*

Values

ErrorCode	Description
SUCCESS	Success
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_PARAM	Parameter error
ERR_NOT_FOUND	Not found
ERR_ABORT	Processing was canceled or near full is detected
ERR_SCAN	Error occurred during the image scan
ERR_NOT_EXEC	Processing is not executed

ImageID

Type

Read/Write*:* *int*

Values

Specifies the ID for an image that is read. Not used for SCNMICRFunctionContinuously and SCNMICRFunctionPostPrint.

Resolution

Type

Readonly: *MfScanDpi*

Values

MfScanDpi	Description
MF_SCAN_DPI_DEFAULT	Scan with default DPI (200 dpi)
MF_SCAN_DPI_100	Scan image at 100 DPI
MF_SCAN_DPI_120	Scan image at 120 DPI
MF_SCAN_DPI_200	Scan image at 200 DPI
MF_SCAN_DPI_240	Scan image at 240 DPI
MF_SCAN_DPI_H240_V200	Horizontal 240 DPI, vertical 200 DPI

AddInfoData

Type

Read/Write*:* *byte[]*

Values

Byte array of additional data to be include in image data.

Image

Type

Readonly: *System.Drawing.Bitmap*

Values

Scanned image interpreted as a rasterized image object.

Data

Type

Readonly: *System.IO.Stream*

Values

A stream of bytes representing the raw image data for the scanned check or ID card.

Status

Type

Readonly: *byte*

Values

Bit values of the byte are as follows.

Bit	Function	Value	
		0	1
0,1,2	Reserved	Fixed to 0	
3	Rescanned	Front	Back
4	Reread	-	Disabled
5	Scanning results	Success	Failure
6	Scanning data overflow	No overflow	Not (Fixed)
7	Scanning data translation error	No error	Ends with an error

Detail

Type

Readonly*:* *byte*

Values

Values of the byte are as follows.

Value	Information
40h	Success
41h	The image scanning result do not exist
42h	Canceling paper insertion standby
44h	Image reading stopped due to a feed error.
45h	A double feed error or an insertion direction error has been occurred during image reading.
46h	A delivery error has been detected before the start of reading.
48h	An error has been detected during the reading operation (paper of irregular length has been inserted, and an error has occurred during paper feed)

MFPrint Class - Properties

This sections contains descriptions of all of the properties exposed by the MFPrint class.

Version

Type

Readonly: *int*

Values

The version of structure used internally within this class.

Ret

Type

Readonly: *ErrorCode*

Values

ErrorCode	Description
SUCCESS	Success
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_TIMEOUT	Unable to complete within timeout
ERR_ACCESS	Unable to read and/or write
ERR_ABORT	Processing was canceled or near full is detected
ERR_LINE_OVERFLOW	A line overflow has occurred during transaction printing
ERR_NOT_EXEC	Processing is not executed

ElectricEndorse

Type

Read/Write*:* *EndorsementType*

Values

EndorsementType	Description
MF_PRINT_TYPE_ELECTRIC_ENDORSE_ONLY	Electronic endorsement only
MF_PRINT_TYPE_ELECTRIC_ENDORSE_EXTEND	Electronic endorsement only Electronic endorsement executed with PrintText and PrintImage

String

Type

Read/Write: StringClass class with indexer of type EndorsementLine for type String

Indexer

indexer	Description
MF_PRINT_LINE_1	First endorsement line
MF_PRINT_LINE_2	Second endorsement line
MF_PRINT_LINE_3	Third endorsement line

Values

String to be output as endorsement line, be sure to include new line character at the end of the string to ensure the next endorsement line appears on the next line.

Attribute

Type

Read/Write: AttributeClass class with indexer of type EndorsementLine for type int

Indexer

indexer	Description
MF_PRINT_LINE_1	First endorsement line
MF_PRINT_LINE_2	Second endorsement line
MF_PRINT_LINE_3	Third endorsement line

Values

Integer value as a binary combination of any of the following values.

Function	Use	Value
Bold	Disable	0
	Enable	1
Underline	None	0
	Single	16
	Double	32
Color	Default Color	0
	Primary Color	256
	Secondary Color	512
	Mixed Both Colors	768
Reversed	Disable	0
	Enable	4096

Font

Type

Read/Write*:* FontClass class with indexer of type EndorsementLine for type FontType

Indexer

indexer	Description
MF_PRINT_LINE_1	First endorsement line
MF_PRINT_LINE_2	Second endorsement line
MF_PRINT_LINE_3	Third endorsement line

Values

FontType	Description
MF_PRINT_FONT_A	Device internal font A
MF_PRINT_FONT_B	Device internal font B

FontSize

Type

Read/Write*:* FontSizeClass class with indexer of type EndorsementLine for type FontScale

Indexer

Indexer	Description
MF_PRINT_LINE_1	First endorsement line
MF_PRINT_LINE_2	Second endorsement line
MF_PRINT_LINE_3	Third endorsement line

Values

FontScale	Description
MF_PRINT_FONT_W1_H1	Normal size
MF_PRINT_FONT_W1_H2	Extra tall font (double height)
MF_PRINT_FONT_W2_H1	Extra wide font (double width)
MF_PRINT_FONT_W2_H2	Double size font (double height and double width)

MFTrueType Class - Methods

This sections contains descriptions of all of the constructors exposed by the MFTrueType class.

MFTrueTypeFont

Construct and instance of this class, setting font parameters if supplied.

Syntax

MFTrueType(*System.Drawing.Font font, PrintColor color, bool reverse*)

MFTrueType(*System.Drawing.Font font, PrintColor color*)

MFTrueType(*System.Drawing.Font font*)

MFTrueType()

Argument

Font:	See Font property for details.
color:	See Color property for details.
reverse:	See Reverse property for details.

MFTrueType Class - Properties

This sections contains descriptions of all of the properties exposed by the MFTrueType class.

Font

Type

Read/Write*:* *System.Drawing.Font*

Values

A System.Drawing.Font object, the font object's Name, SizeInPoints and Bold and Underline properties are used and maintained.

Notes

All properties of the SystemDrawing.Font object are ignored other than Name, SizeInPoints, Bold and Underline.

Color

Type

Read/Write*:* *PrintColor*

Values

PrintColor	Description
MF_PRINT_DEFAULT	Default color
MF_PRINT_BLACK	Primary color
MF_PRINT_COLOR	Secondary color
MF_PRINT_MIXED	Mix of primary and secondary colors

Reverse

Type

Read/Write*:* *bool*

Values

bool	Description
true	Typeface is reversed
false	Typeface is not reversed

MFProcess Class - Properties

This sections contains descriptions of all of the properties exposed by the MFProcess class.

Version

Type

Readonly: *int*

Values

The version of structure used internally within this class.

ActivationMode

Type

Read/Write*:* *MfActivateMode*

Values

MfActivateMode	Description
MF_ACTIVATE_MODE_HIGH_SPEED	Execute processing in High Speed mode
MF_ACTIVATE_MODE_CONFIRMATION	Execute processing in Confirmation mode

PaperType

Type

Read/Write*:* *MfPaperType*

Values

MfPaperType	Description
MF_PAPER_TYPE_CHECK	Scan only check paper
MF_PAPER_TYPE_OTHER	Scan check paper, stub or any other type of paper

Notes

When MF_PAPER_TYPE_CHECK is specified and a paper other than check papers is scanned, a scan error may occur.
When MF_PAPER_TYPE_OTHER is selected, ease the double feed detection level.
The threshold for detecting double feed can be changed using registry or setting file equivalent to the registry. The default value is listed below.

MfPaperType	Default value
MF_PAPER_TYPE_CHECK	0.17 mm
MF_PAPER_TYPE_OTHER	0.23 mm

StartWaitTime

Type

Read/Write*:* *uint*

Values

Specify a time interval from when paper on the ASF is detected and scanning the paper is started.
Settable values are 0 to 6400 in units of ms. The default is 1000.
A value larger than 6400 is rounded down to 6400.

SuccessStamp

Type

Read/Write*:* *MfStamp*

Values

MfStamp	Description
MF_STAMP_ENABLE	Enable a stamp when scan is completed successfully
MF_STAMP_DISABLE	Disable a stamp when scan is completed successfully

Notes

This stamp setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this stamp setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

PaperMisInsertionErrorSelect

Type

Read/Write*:* *MfErrorSelect*

Values

MfErrorSelect	Description
MF_ERROR_SELECT_NODETECT	Not detect a paper insertion error
MF_ERROR_SELECT_DETECT	Detect a paper insertion error

Notes

When not detecting the error is selected, all the settings of PaperMisInsertionStamp, PaperMisInsertionErrorEject, and PaperMisInsertionCancel are ignored.

PaperMisInsertionErrorEject

Type

Read/Write*:* *MfEject*

Values

MfEject	Description
MF_EJECT_MAIN_POCKET	When a paper insertion error occurs, eject the paper to the main pocket
MF_EJECT_SUB_POCKET	When a paper insertion error occurs, eject the paper to the sub pocket
MF_EJECT_NOEJECT	When a paper insertion error occurs, not eject the paper

Notes

When MF_ERROR_SELECT_NODETECT is specified to PaperMisInsertionErrorSelect, this PaperMisInsertionErrorEject setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

PaperMisInsertionStamp

Type

Read/Write*:* *MfStamp*

Values

MfStamp	Description
MF_STAMP_ENABLE	When a paper insertion error occurs, enable a stamp
MF_STAMP_DISABLE	When a paper insertion error occurs, disable a stamp

Notes

When MF_ERROR_SELECT_NODETECT is specified to PaperMisInsertionErrorSelect, this PaperMisInsertionStamp setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

PaperMisInsertionCancel

Type

Read/Write*:* *MfCancel*

Values

MfCancel	Description
MF_CANCEL_DISABLE	When a paper insertion error occurs, scanning the paper is not canceled
MF_CANCEL_ENABLE	When a paper insertion error occurs, scanning the paper is canceled

Notes

When MF_ERROR_SELECT_NODETECT is specified to PaperMisInsertionErrorSelect, this PaperMisInsertionCancel setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

NoiseErrorSelect

Type

Read/Write: *MfErrorSelect*

Values

MfErrorSelect	Description
MF_ERROR_SELECT_NODETECT	Not detect an external noise error
MF_ERROR_SELECT_DETECT	Detect an external noise error

Notes

When not detecting the error is selected, all the settings of NoiseErrorEject, NoiseErrorStamp and NoiseErrorCancel are ignored.

NoiseErrorEject

Type

Read/Write: *MfEject*

Values

MfEject	Description
MF_EJECT_MAIN_POCKET	Eject paper to the main pocket when an external noise error occurs
MF_EJECT_SUB_POCKET	Eject paper to the sub pocket when an external noise error occurs
MF_EJECT_NOEJECT	Not eject paper when an external noise error occurs

Notes

When MF_ERROR_SELECT_NODETECT is specified to NoiseErrorSelect, this NoiseErrorEject setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

NoiseStamp

Type

Read/Write*:* *MfStamp*

Values

MfStamp	Description
MF_STAMP_ENABLE	Enable a stamp when an external noise error occurs
MF_STAMP_DISABLE	Disable a stamp when an external noise error occurs

Notes

When MF_ERROR_SELECT_NODETECT is specified to NoiseErrorSelect, this NoiseStamp setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

NoiseCancel

Type

Read/Write*:* *MfCancel*

Values

MfCancel	Description
MF_CANCEL_DISABLE	Scanning is not canceled when an external noise error occurs
MF_CANCEL_ENABLE	Scanning is canceled when an external noise error occurs

Notes

When MF_ERROR_SELECT_NODETECT is specified to NoiseErrorSelect, the NoiseCancel setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

DoubleFeedErrorSelect

Type

Read/Write*:* *MfErrorSelect*

Values

MfErrorSelect	Description
MF_ERROR_SELECT_NODETECT	Not detect a double feed error
MF_ERROR_SELECT_DETECT	Detect a double feed error

Notes

When not detecting the error is selected, all the settings of DoubleFeedErrorEject, DoubleFeedStamp, and DoubleFeedCancel are ignored.

DoubleFeedErrorEject

Type

Read/Write*:* *MfEject*

Values

MfEject	Description
MF_EJECT_MAIN_POCKET	When a double feed error occurs, eject the paper to the main pocket
MF_EJECT_SUB_POCKET	When a double feed error occurs, eject the paper to the sub pocket
MF_EJECT_NOEJECT	When a double feed error occurs, not eject the paper

Notes

When MF_ERROR_SELECT_NODETECT is specified to DoubleFeedErrorSelect, this DoubleFeedErrorEject setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

DoubleFeedStamp

Type

Read/Write*:* *MfStamp*

Values

MfStamp	Description
MF_STAMP_ENABLE	Enable a stamp when a double feed error occurs
MF_STAMP_DISABLE	Disable a stamp when a double feed error occurs

Notes

When MF_ERROR_SELECT_NODETECT is specified to DoubleFeedErrorSelect, this DoubleFeedStamp setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

DoubleFeedCancel

Type

Read/Write*:* *MfCancel*

Values

MfCancel	Description
MF_CANCEL_DISABLE	Scanning is not canceled when a double feed error occurs
MF_CANCEL_ENABLE	Scanning is canceled when a double feed error occurs

Notes

When MF_ERROR_SELECT_NODETECT is specified to DoubleFeedErrorSelect, this DoubleFeedCancel setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.

BaddataErrorSelect

Type

Read/Write*:* *MfErrorSelect*

Values

MfErrorSelect	Description
MF_ERROR_SELECT_NODETECT	Not detect a MICR character recognition error
MF_ERROR_SELECT_DETECT	Detect a MICR character recognition error

Notes

When not detecting the error is selected, all the settings of BaddataCount, BaddataErrorEject, BaddataStamp, and BaddataCancel are cancelled.

BaddataCount

Type

Read/Write*:* *byte*

Values

Limit the allowable number of unanalyzable characters.
Specifying zero allows no unanalyzable characters.
When a number (1 to 254) is specified, the specified number of unanalyzable characters are allowed.

Notes

When MF_ERROR_SELECT_NODETECT is specified to BaddataErrorSelect, this BaddataCount setting is ignored.
The settings of BaddataErrorEject, BaddataStamp, and BaddataCancel are enabled when the number of detected unanalyzable characters exceed the number specified by this setting.

BaddataErrorEject

Type

Read/Write*:* *MfEject*

Values

MfEject	Description
MF_EJECT_MAIN_POCKET	Eject paper to the main pocket when the number of unanalyzable characters exceed the specified limit
MF_EJECT_SUB_POCKET	Eject paper to the sub pocket when the number of unanalyzable characters exceed the specified limit
MF_EJECT_NOEJECT	Not eject paper when the number of unanalyzable characters exceed the specified limit

Notes

When MF_ERROR_SELECT_NODETECT is specified to BaddataErrorSelect, or when the number of unanalyzable characters does not exceed the limit specified with the BaddataCount, this BaddataErrorEject setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.
The scanning speed becomes slower when other than MF_EJECT_MAIN_POCKET is selected and MF_ACTIVATE_MODE_HIGH_SPEED is specified to ActivationMode.

BaddataStamp

Type

Read/Write*:* *MfStamp*

Values

MfStamp	Description
MF_STAMP_DISABLE	Disable a stamp when the number of unanalyzable characters exceed the specified limit
MF_STAMP_ENABLE	Enable a stamp when the number of unanalyzable characters exceed the specified limit

Notes

When MF_ERROR_SELECT_NODETECT is specified to BaddataErrorSelect, or when the number of unanalyzable characters does not exceed the limit specified with the BaddataCount, this BaddataStamp setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.
The scanning speed becomes slower when this setting conflicts with settings of SuccessStamp and NodataStamp while MF_ACTIVATE_MODE_HIGH_SPEED is specified to ActivationMode.

BaddataCancel

Type

Read/Write*:* *MfCancel*

Values

MfCancel	Description
MF_CANCEL_DISABLE	Scanning is not cancelled when the number of unanalyzable characters exceed the specified limit
MF_CANCEL_ENABLE	Scanning is cancelled when the number of unanalyzable characters exceed the specified limit

Notes

When MF_ERROR_SELECT_NODETECT is specified to BaddataErrorSelect, or when the number of unanalyzable characters does not exceed the limit specified with the BaddataCount, this BaddataCancel setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.
The scanning speed becomes slower when MF_CANCEL_ENABLE is selected while MF_ACTIVATE_MODE_HIGH_SPEED is specified to ActivationMode.

NodataErrorSelect

Type

Read/Write*:* *MfErrorSelect*

Values

MfErrorSelect	Description
MF_ERROR_SELECT_DETECT	Cause an error when MICR magnetic waveform cannot be detected
MF_ERROR_SELECT_NODETECT	Not cause an error when MICR magnetic waveform cannot be detected

Notes

When not causing the error is selected, all the settings of NodataErrorEject, NodataStamp, and NodataCancel are ignored.

NodataErrorEject

Type

Read/Write*:* *MfEject*

Values

MfEject	Description
MF_EJECT_MAIN_POCKET	Eject paper to the main pocket when a MICR magnetic waveform error occurs
MF_EJECT_SUB_POCKET	Eject paper to the sub pocket when a MICR magnetic waveform error occurs
MF_EJECT_NOEJECT	Not eject paper when a MICR magnetic waveform error occurs

Notes

When MF_ERROR_SELECT_NODETECT is specified to NodataErrorSelect, this NodataErrorEject setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.
The scanning speed becomes slower when other than MF_EJECT_MAIN_POCKET is selected while MF_ACTIVATE_MODE_HIGH_SPEED is specified to ActivationMode.

NodataStamp

Type

Read/Write*:* *MfStamp*

Values

MfStamp	Description
MF_STAMP_DISABLE	Disable a stamp when a MICR magnetic waveform error occurs
MF_STAMP_ENABLE	Enable a stamp when a MICR magnetic waveform error occurs

Notes

When MF_ERROR_SELECT_NODETECT is specified to NodataErrorSelect, this NodataStamp setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.
The scanning speed becomes slower when this setting conflicts with settings of SuccessStamp and BaddataStamp while MF_ACTIVATE_MODE_HIGH_SPEED is specified to ActivationMode.

NodataCancel

Type

Read/Write*:* *MfCancel*

Values

MfCancel	Description
MF_CANCEL_DISABLE	Scanning is not cancelled when a MICR magnetic waveform error occurs
MF_CANCEL_ENABLE	Scanning is cancelled when a MICR magnetic waveform error occurs

Notes

When MF_ERROR_SELECT_NODETECT is specified to NodataErrorSelect, this NodataCancel setting is ignored.
This setting is applied with no exceptions in High Speed mode.
In Confirmation mode, this setting is applied only when SetBehaviorToScnResult is not called in the MF_DATARECEIVE_DONE callback.
The scanning speed becomes slower when MF_CANCEL_ENABLE is selected while MF_ACTIVATE_MODE_HIGH_SPEED is specified to ActivationMode.

NearFullSelect

Type

Read/Write*:* *MfNearFullSelect*

Values

MfNearFullSelect	Description
MF_NEARFULL_PERMIT	Allow scanning operation when the pocket is nearly full
MF_NEARFULL_NOT_PERMIT	Not allow scanning operation when the pocket is nearly full

ResultPartialData

Type

Read/Write*:* *MfResult*

Values

MfResult	Description
MF_RESULT_NONE	When an error occurs in the middle of scanning, the imperfect scanned data is not returned to the host processor
MF_RESULT_PARTIAL	When an error occurs in the middle of scanning, the imperfect scanned data is returned to the host processor

MFOcrAb Class - Properties

This sections contains descriptions of all of the properties exposed by the MFOcrAb class.

Version

Type

Readonly: *int*

Values

The version of structure used internally within this class.

Ret

Type

Readonly: *ErrorCode*

Values

One of the following values is returned.

ErrorCode	Description
SUCCESS	Success
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_FOUND	Not found
ERR_IMAGE_FILEOPEN	Open failure
ERR_EXEC_FUNCTION	Other API is running
ERR_MICR_NODATA	MICR data is not existing

OcrType

Type

Read/Write*:* *MfOcrType*

Values

MfOcrType	Description
MF_OCR_FONT_OCRA_NUM	Numeric characters in OCR-A font
MF_OCR_FONT_OCRB_NUM	Numeric characters in OCR-B font
MF_OCR_FONT_OCRA_ALPHA	Alphabetic characters in OCR-A font
MF_OCR_FONT_OCRB_ALPHA	Alphabetic characters in OCR-B font
MF_OCR_FONT_OCRA_ALPHANUM	Alphanumeric characters in OCR-A font
MF_OCR_FONT_OCRB_ALPHANUM	Alphanumeric characters in OCR-B font
MF_OCR_FONT_OCRA_ALPHANUM_WOOH	Alphanumeric characters in OCR-A font (except OH)
MF_OCR_FONT_OCRB_ALPHANUM_WOOH	Alphanumeric characters in OCR-B font (except OH)
MF_OCR_FONT_OCRA_ALPHANUM_WOZERO	Alphanumeric characters in OCR-A font (except ZERO)
MF_OCR_FONT_OCRB_ALPHANUM_WOZERO	Alphanumeric characters in OCR-B font (except ZERO)
MF_OCR_FONT_OCRA_SYMNUM	OCR-A font, numeric numbers and symbols ("+" excluded)
MF_OCR_FONT_OCRB_SYMNUM	OCR-B font, numeric numbers and symbols ("+" included)

Direction

Type

Read/Write*:* *MfDirection*

Values

MfDirection	Description
MF_OCR_LEFTRIGHT	From left to right (normal direction)
MF_OCR_TOPBOTTOM	From top to bottom (rotated 90 degrees clockwise)
MF_OCR_RIGHTLEFT	From right to left (upside-down)
MF_OCR_BOTTOMTOP	From bottom to top (rotated 90 degrees counterclockwise)

StartX

Type

Read/Write*:* *uShort*

Values

Specify start X coordinate of the OCR recognition area. (unit: mm)
The allowable range is between 0 and 254.
A parameter error will be returned if a value outside the range is specified.

StartY

Type

Read/Write: *uShort*

Values

Specify start Y coordinate of the OCR recognition area. (unit: mm)
The allowable range is between 0 and 254.
A parameter error will be returned if a value outside the range is specified.

EndX

Type

Read/Write: *uShort*

Values

Specify end X coordinate of the OCR recognition area. (unit: mm)
The settable values are 1 through 255, OCR_AREA_RIGHT, and OCR_AREA_LEFT.
When OCR_AREA_RIGHT is specified, the right side of an image can be specified.
When OCR_AREA_LEFT is specified, the left side of an image can be specified.
In the following cases, the parameter error (ERR_PARAM) is returned:

- When a value other than OCR_AREA_RIGHT or OCR_AREA_LEFT is specified in the specified reading range and StartX ≥ EndX is set
- When Direction is set to MF_OCR_TOPBOTTOM or MF_OCR_BOTTOMTOP
- When the specified reading range is outside 1 ≤ (EndX - StartX) ≤ 14

EndY

Type

Read/Write: *uShort*

Values

Specify end Y coordinate of the OCR recognition area. (unit: mm)
The settable values are 1 through 255, OCR_AREA_BOTTOM, and OCR_AREA_TOP.
If OCR_AREA_BOTTOM is specified when OCR origin is set to top left or top right, the bottom of the image can be specified.
If OCR_AREA_TOP is specified when OCR origin is set to bottom left or bottom right, the top of the image can be specified.
In the following cases, the parameter error (ERR_PARAM) is returned:

- when a value other than OCR_AREA_BOTTOM or OCR_AREA_TOP is specified in the specified reading range and StartY ≥ EndY is set
- when Direction is set to MF_OCR_LEFTRIGHT or MF_OCR_RIGHTLEFT
- when the reading reange is outside 1 ≤ (EndY - StartY) ≤ 14

SpeceHandling

Type

Read/Write*:* *MfSpeceHandling*

Values

MfSpeceHandling	Description
OCR_SPACE_ENABLE	Include space characters in OCR recognition result
OCR_SPACE_DISABLE	Not include space characters in OCR recognition result

OcrStr

Type

Readonly: *String*

Values

Store character strings acquired by OCR recognition process.

OcrReliableInfo.FirstSelectString

Type

Readonly: *String*

Values

Best guess raw data read optically from OCR characters.

OcrReliableInfo.SecondSelectString

Type

Readonly: *String*

Values

Second guess raw data read optically from OCR characters.

OcrReliableInfo.FirstSelectPercentage

Type

Readonly: *int[]*

Values

Confidence level of best guess raw data read optically from OCR characters.

OcrReliableInfo.SecondSelectPercentage

Type

Readonly: *int[]*

Values

Confidence level of second guess raw data read optically from OCR characters.

MFVersion Class - Properties

This sections contains descriptions of all of the properties exposed by the MFVersion class.

Description

Type

Readonly: *String*

Values

Target information to be acquired is set.

Version

Type

Readonly: *String*

Values

Acquired version information is set.

MFIqa Class - Properties

This sections contains descriptions of all of the properties exposed by the MFIqa class.

Version

Type

Readonly: *int*

Values

The version of structure used internally within this class.

ErrorSelect

Type

Read/Write*:* *MfErrorSelect*

Values

MfErrorSelect	Description
MF_ERROR_SELECT_DETECT	Detects an error when document does not pass the IQA validation.
MF_ERROR_SELECT_NODETECT	Does not detects an error when document does not pass the IQA validation. The Settings of ErrorEject, Stamp and Cancel are ignored.

ErrorEject

Type

Read/Write*:* MfEject

Values

MfEject	Description
MF_EJECT_MAIN_POCKET	Ejects to the main pocket when IQA validation error occurs.
MF_EJECT_SUB_POCKET	Ejects to the sub pocket when IQA validation error occurs.
MF_EJECT_NOEJECT	Does not eject the document when IQAvalidation error occurs.

Notes

- ❏ When MF_ERROR_SELECT_NODETECT is set for ErrorSelect, this setting is ignored.
- ❏ In the Confirmation mode, the setting for SetBehaviorToScnResult has the priority.
- ❏ When ActivationMode of MFProcess Class is set to MF_ACTIVATE_MODE_HIGH_SPEED, and a setting other than MF_EJECT_MAIN_POCKET is set, the reading speed is lowered.

Stamp

Type

Read/Write: *MfStamp*

Values

MfStamp	Description
MF_STAMP_DISABLE	Does not execute franking when the IQA validation error occurs.
MF_STAMP_ENABLE	Executes franking when the IQA validation error occurs.

Notes

- ❏ When MF_ERROR_SELECT_NODETECT is set for ErrorSelect, this setting is ignored.
- ❏ In the Confirmation mode, the setting for SetBehaviorToScnResult has the priority.
- ❏ When this setting is different from that of SuccessStamp for MFProcess Class, the reading speed is lowered.

Cancel

Type

Read/Write: *MfCancel*

Values

MfCancel	Description
MF_CANCEL_DISABLE	Does not cancel the reading process when the IQA validation error occurs.
MF_CANCEL_ENABLE	Cancels the reading process when the IQA validation error occurs.

Notes

- ❏ When MF_ERROR_SELECT_NODETECT is set for ErrorSelect, this setting is ignored.
- ❏ In the Confirmation mode, the setting for SetBehaviorToScnResult has priority.
- ❏ When ActivationMode of MFProcess Class is set to MF_ACTIVATE_MODE_HIGH_SPEED, and MF_CANCEL_ENABLE is set for this setting, the reading speed is lowered.

ImageFormat

Type

Read/Write: *Format*

Values

Specifies the image format for the IQA validation. For details of the setting value, see "SCNSetImageFormat / SCNGetImageFormat" on page 4-36.

ColorDepth

Type

Read/Write: *ColorDepth*

Values

Specifies the gradation for the IQA validation. For details of the setting value, see "SCNSetImageQuality / SCNGetImageQuality" on page 4-34.

Threshold

Type

Read/Write: *double*

Values

Specifes the density threshold for the IQA validation. Specify when the gradation is Black and White. For details of the setting value, see "SCNSetImageQuality / SCNGetImageQuality" on page 4-34.

Color

Type

Read/Write: *Color*

Values

Specifies the color for the IQA validation. For details of the setting value, see "SCNSetImageQuality / SCNGetImageQuality" on page 4-34.

ExOption

Type

Read/Write: *ExOption*

Values

Specifies the variety of density adjustment for the IQA validation. For details of the setting value, see "SCNSetImageQuality / SCNGetImageQuality" on page 4-34.

Resolution

Type

Read/Write: *MfScanDpi*

Values

Specifies the resolution for the IQA validation. For details of the setting value, see "Resolution" on page 4-112.

Undersize

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the UndersizeImage validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the UndersizeImage validation.
MF_IQA_TEST_ENABLEE	Enables the UndersizeImage validation.

Oversize

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the OversizeImage validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the OversizeImage validation.
MF_IQA_TEST_ENABLEE	Enables the OversizeImage validation.

Mincompressed

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the MinCompressedImageSize validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the MinCompressedImageSize validation.
MF_IQA_TEST_ENABLEE	Enables the MinCompressedImageSize validation.

Notes

When the following settings are not made for ColorDepth and ImageFormat, the MinCompressedImageSize validation is disabled even when MF_IQA_TEST_ENABLEE is set.

ColorDepth setting	ImageFormat setting
EPS_BI_SCN_1BIT	EPS_BI_SCN_TIFF
	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
EPS_BI_SCN_8BIT	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF

Maxcompressed

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the MaxCompressedImageSize validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the MaxCompressedImageSize validation.
MF_IQA_TEST_ENABLEE	Enables the MaxCompressedImageSize validation.

Notes

When the following settings are not made for ColorDepth and ImageFormat, the MaxCompressedImageSize validation is disabled even when MF_IQA_TEST_ENABLEE is set.

ColorDepth setting	ImageFormat setting
EPS_BI_SCN_1BIT	EPS_BI_SCN_TIFF
	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
EPS_BI_SCN_8BIT	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF

Front_rear

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the FrontRearImageMismatch validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the FrontRearImageMismatch validation.
MF_IQA_TEST_ENABLEE	Enables the FrontRearImageMismatch validation.

Toolight

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the ImageTooLight validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the ImageTooLight validation.
MF_IQA_TEST_ENABLEE	Enables the ImageTooLight validation.

Toodark

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the ImageTooDark validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the ImageTooDark validation.
MF_IQA_TEST_ENABLEE	Enables the ImageTooDark validation.

Streaks

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the HorizontalStreaksPresent validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the HorizontalStreaksPresent validation.
MF_IQA_TEST_ENABLEE	Enables the HorizontalStreaksPresent validation.

Noise

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the ExcessiveSpotNoise validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the ExcessiveSpotNoise validation.
MF_IQA_TEST_ENABLEE	Enables the ExcessiveSpotNoise validation.

Notes

When ColorDepth is set to 256 gray scale, the ExcessiveSpotNoise validation is disabled even when this setting is set to MF_IQA_TEST_ENABLE.

Focus

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the ImageOutOfFocus validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the ImageOutOfFocus validation.
MF_IQA_TEST_ENABLEE	Enables the ImageOutOfFocus validation.

Notes

When ColorDepth is set to Black and White, the ImageOutOfFocus validation is disabled even when this setting is set to MF_IQA_TEST_ENABLE.

Corners

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the FoldedTornDocCorners validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the FoldedTornDocCorners validation.
MF_IQA_TEST_ENABLEE	Enables the FoldedTornDocCorners validation.

Edges

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the FoldedTornDocEdges validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the FoldedTornDocEdges validation.
MF_IQA_TEST_ENABLEE	Enables the FoldedTornDocEdges validation.

Framing

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the DocFramingError validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the DocFramingError validation.
MF_IQA_TEST_ENABLEE	Enables the DocFramingError validation.

Skew

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disable the ExcessiveDocSkew validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the ExcessiveDocSkew validation.
MF_IQA_TEST_ENABLEE	Enables the ExcessiveDocSkew validation.

Carbon

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the CarbonStripDetection validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the CarbonStripDetection validation.
MF_IQA_TEST_ENABLEE	Enables the CarbonStripDetection validation.

Piggyback

Type

Read/Write*:* *MfIqaTest*

Values

Enables/Disables the Piggyback validation.

MfCancel	Description
MF_IQA_TEST_DISABLE	Disables the Piggyback validation.
MF_IQA_TEST_ENABLEE	Enables the Piggyback validation.

Explanation

The Piggyback validation is executed from the double feed detection result of the device.

MFIqaResult Class - Properties

This section contains descriptions of all of the properties exposed by the MFIqaResult class.

Version

Type

Readonly: *int*

Values

The version of structure used internally within this class.

Ret

Type

Readonly: *ErrorCode*

Values

One of the following values is returned.

ErrorCode	Description
SUCCESS	Success
ERR_SCAN	Error occurred during the image scan
ERR_SCAN_IQA	Detects an error at the IQA validation.

UndersizeImage

Sets the UndersizeImage validation result.

Type

Readonly*:* *IqaResultUnderSizeImageClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

Width: Sets the horizontal size (unit: 0.1 inch) of an image.

Height: Sets the vertical size (unit: 0.1 inch) of an image.

OversizeImage

Sets the Oversize validation result.

Type

Readonly*:* *IqaResultOverSizeImageClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

Width: Sets the horizontal size (unit: 0.1 inch) of an image.

Height: Sets the vertical size (unit: 0.1 inch) of an image.

MaxCompressedImageSize

Sets the MaxCompressed validation result.

Type

Readonly*:* *IqaResultMaxCompressedImageSizeClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iSize: Sets the compression size (unit: Byte).

MinCompressedImageSize

Sets the MinCompressed validation result.

Type

Readonly*:* *IqaResultMinCompressedImageSizeClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iSize: Sets the compression size (unit: Byte).

FrontRearImageMismatch

Sets the FrontRearImageMismatch validation result.

Type

Readonly: *IqaResultFrontRearImageMismatchClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iAbsWidthDiff: Sets the horizontal size difference between front and back image (unit: 0.1 inch).

iAbsHeightDiff: Sets the vertical size difference between front and back image (unit: 0.1 inch).

ImageTooLight

Sets the ImageTooLight validation result.

Type

Readonly: *IqaResultImageTooLightClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iBlackPixels: Sets the black pixel percentage (unit: 0.1%) of an image.

iBrightness: Sets the brightness (unit: 0.1%) of an image.

iContrast: Sets the contrast (unit: 0.1%) of an image.

ImageTooDark

Sets the ImageTooDark validation result.

Type

Readonly: *IqaResultImageTooDarkClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iBlackPixels: Sets the black pixel percentage (unit: 0.1%) of an image.

iBrightness: Sets the brightness (unit: 0.1%) of an image.

HorizontalStreaksPresent

Sets the HorizontalStreaksPresent validation result.

Type

Readonly*:* *IqaResultHorizontalStreaksPresentClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iStreakCount: Sets the number of black lines in a Black and White image.

iStreakHeight: Sets the width of the thickest black line.

ExcessiveSpotNoise

Sets the SpotNosise validation result.

Type

Readonly*:* *IqaResultExcessiveSpotNoiseClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iCount: Sets the average number of spots (noise) per square inch.

ImageOutOfFocus

Sets the OutOfFocus validation result. This validation is for 256 GrayScale images.

Type

Readonly*:* *IqaResultImageOutOfFocusClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iImageFocusScore: Sets the score from the following calculation formula.
(max video gradient) / (gray level dynamic range) * (pixel pitch)

FoldedTornDocCorners

Sets the FoldedTornCorner validation result.

Type

Readonly*:* *IqaResultFoldedTornDocCornersClass*

Properties

Result:	Sets the validation result. For details, see "GetIqaResult" on page 4-93.
iTopLeftWidth:	Sets the horizontal width (unit: 0.1 inch) of a torn or folded part on the top left corner of an image.
iTopLeftHeight:	Sets the vertical width (unit: 0.1 inch) of a torn or folded part on the top left corner of an image.
iTopRightWidth:	Sets the horizontal width (unit: 0.1 inch) of a torn or folded part on the top right corner of an image.
iTopRightHeight:	Sets the vertical width (unit: 0.1 inch) of a torn or folded part on the top right corner of an image.
iBottomLeftWidth:	Sets the horizontal width (unit: 0.1 inch) of a torn or folded part on the bottom left corner of an image.
iBottomLeftHeight:	Sets the vertical width (unit: 0.1 inch) of a torn or folded part on the bottom left corner of an image.
iBottomRightWidth:	Sets the horizontal width (unit: 0.1 inch) of a torn or folded part on the bottom right corner of an image.
iBottomRightHeight:	Sets the vertical width (unit: 0.1 inch) of a torn or folded part on the bottom right corner of an image.

FoldedTornDocEdges

Sets the FoldedTornEdge validation result.

Type

Readonly*:* *IqaResultFoldedTornDocEdgesClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iTopWidth: Sets the horizontal width (unit: 0.1 inch) of a torn or folded part on the top side of an image.

iTopHeight: Sets the vertical width (unit: 0.1 inch) of a torn or folded part on the top side of an image.

iLeftWidth: Sets the horizontal width (unit: 0.1 inch) of a torn or folded part on the left side of an image.

iLeftHeight: Sets the vertical width (unit: 0.1 inch) of a torn or folded part on the left side of an image.

iRightWidth: Sets the horizontal width (unit: 0.1 inch) of a torn or folded part on the right side of an image.

iRightHeight: Sets the vertical width (unit: 0.1 inch) of a torn or folded part on the right side of an image.

iBottomWidth: Sets the horizontal width (unit: 0.1 inch) of a torn or folded part on the bottom side of an image.

iBottomHeight: Sets the vertical width (unit: 0.1 inch) of a torn or folded part on the bottom side of an image.

DocFramingError

Sets the FramingError validation result.

Type

Readonly*:* *IqaResultDocFramingErrorClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iTop: Sets the margin width (unit: 0.1 inch) on the top side of an image.

iLeft: Sets the margin width (unit: 0.1 inch) on the left side of an image.

iRight: Sets the margin width (unit: 0.1 inch) on the left side of an image.

iBottom: Sets the margin width (unit: 0.1 inch) on the bottom side of an image.

ExcessiveDocSkew

Sets the ExcessiveSkew validation result.

Type

Readonly: *IqaResultExcessiveDocSkewClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iAngle: Sets the tilt angle (unit: 0.1°).

iRange: Fixed to 0.

CarbonStripDetection

Sets the CarbonStripDetection validation result.

Type

Readonly: *IqaResultCarbonStripDetectionClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

iAngle: Sets the height (unit: 0.1 inch) of the carbon strip.

Piggyback

Sets the Piggyback validation result.

Type

Readonly: *IqaResultPiggybackClass*

Properties

Result: Sets the validation result. For details, see "GetIqaResult" on page 4-93.

MFBarcode Class - Properties

This section contains descriptions of all of the properties exposed by the MFBarcode class.

Version

Obtains the version of this class.

Type

Readonly: *int*

Values

The version of structure used internally within this class.

Ret

Obtains the execution result of barcodes.

Type

Readonly: *ErrorCode*

ErrorCode	Description
SUCCESS	Success
ERR_NO_MEMORY	Memory not allocated or could not allocate
ERR_HANDLE	Not opened correctly
ERR_PARAM	Parameter error
ERR_NOT_FOUND	Not found
ERR_IMAGE_FILEOPEN	Open failure
ERR_BARCODE_NODA	Barcode cannot be detected.

ErrorSelect

Specifies whether to detect the barcode decode error.

Type

Read/Write*:* *MfErrorSelect*

Values

MfErrorSelect	Description
MF_ERROR_SELECT_DETECT	Detects errors.
MF_ERROR_SELECT_NODETECT	Not detects errors.

Notes

If MF_ERROR_SELECT_DETECT is specified, the settings of ErrorEject, Stamp, Cancel are ignored.

ErrorEject

Specifies where to eject when the barcode decode error is detected.

Type

Read/Write: MfEject

Values

MfEject	Description
MF_EJECT_MAIN_POCKET	Ejects documents to the main pocket.
MF_EJECT_SUB_POCKET	Ejects documents to the sub pocket.
MF_EJECT_NOEJECT	Not ejects documents.

Notes

- ❏ When MF_ERROR_SELECT_NODETECT is set for ErrorSelect, this setting is ignored.
- ❏ In the Confirmation mode, the setting for SetBehaviorToScnResult has the priority.
- ❏ When ActivationMode of MFProcess Class is set to MF_ACTIVATE_MODE_HIGH_SPEED, and a setting other than MF_EJECT_MAIN_POCKET is set, the reading speed is lowered.

Stamp

Specifies whether to do the flanking process to the sheet when the barcode decode error is detected.

Type

Read/Write: *MfStamp*

Values

MfStamp	Description
MF_STAMP_DISABLE	Does the flanking process.
MF_STAMP_ENABLE	Not does the flanking process.

Notes

- ❏ When MF_ERROR_SELECT_NODETECT is set for ErrorSelect, this setting is ignored.
- ❏ In the Confirmation mode, the setting for SetBehaviorToScnResult has the priority.
- ❏ When this setting is different from that of SuccessStamp for MFProcess Class, the reading speed is lowered.

Cancel

Specifies whether to continue the reading process when the barcode decode error is detected.

Type

Read/Write*:* *MfCancel*

Values

MfCancel	Description
MF_CANCEL_DISABLE	Continues the reading process.
MF_CANCEL_ENABLE	Not continues the reading process.

Notes

- ❏ When MF_ERROR_SELECT_NODETECT is set for ErrorSelect, this setting is ignored.
- ❏ In the Confirmation mode, the setting for SetBehaviorToScnResult has priority.
- ❏ When ActivationMode of MFProcess Class is set to MF_ACTIVATE_MODE_HIGH_SPEED, and MF_CANCEL_ENABLE is set for this setting, the reading speed is lowered.

TargetColor

Specifies the barcode color.

Type

Read/Write*:* *MfBarcodeTargetColor*

Values

MfBarcodeTargetColor	Description
MF_BARCODE_TARGET_COLOR_GRAY	Decodes the gray scale images.

Resolution

Specifies the image resolution of image data.

Type

Read/Write*:* *MfScanDpi*

Values

MfScanDpi	Description
MF_SCAN_DPI_DEFAULT	200 dpi

InfoMode

Specifies the Index of the element to refer to, which is set in the Info property.

Type

Read/Write*:* *int*

Values

Index	Description
0	Refers to the setting of element "zero" in the arrangement of Info property. Refers only to SymbolMask property and Direction property.

Info

The decode setting and the status of decode result are set in MFBarcode.BarcodeInfo Class.

Type

Readonly*:* *MFBarcode.BarcodeInfo[5]*

MFBarcode.BarcodeInfo Class

BarcodeInfo.Ret

The barcode decode execution result is set.

Type

Readonly*:* *ErrorCode*

Values

ErrorCode	Description
SUCCESS	Success
ERR_BARCODE_NODATA	Barcode cannot be detected.

BarcodeInfo.Status

The status of decoding result is set.

Type

Readonly*:* *byte*

Values

Bit	Function	Value	
		0	1
0,1,2	Reserved	Fixed to 0	
3	Detailed information	-	Added
4	Reserved	Fixed to 0	
5	Reading result	Success	Failure
6,7	Reserved	Fixed to 0	

BarcodeInfo.Detail

The detailed status of decoding result is set.

Type

Readonly*:* *byte*

Values

Value	Information
40h	Success
45h	Barcode cannot be detected.

BarcodeInfo.SymbolMask

Specifies the barcode symbol to decode.

Type

Read/Write*:* *MfBarcodeSymbol*

Values

MfBarcodeSymbol	Barcode
MF_BARCODE_SYMBOL_CODABAR	Codabar
MF_BARCODE_SYMBOL_CODE128	Code128
MF_BARCODE_SYMBOL_CODE39	Code39
MF_BARCODE_SYMBOL_ITF	ITF
MF_BARCODE_SYMBOL_EAN_JAN	JAN13(EAN), JAN8(EAN)
MF_BARCODE_SYMBOL_UPC_A	UPC-A
MF_BARCODE_SYMBOL_UPC_E	UPC-E

Explanation

Multiple barcode symbols can be specified. In such case, specify them with a logical sum of each symbol value.

BarcodeInfo.Direction

Sets the decode direction.

Type

Read/Write*:* *MfBarcodeDirection*

Values

MfBarcodeSymbol	Barcode
MF_BARCODE_DIRECTION_ALL	Decodes it in both directions from left to right and top to bottom
MF_BARCODE_DIRECTION_LEFTRIGHT	Decodes it from left to right
MF_BARCODE_DIRECTION_TOPBOTTOM	Decodes it from top down
MF_BARCODE_DIRECTION_RIGHTLEFT	Decodes it from right to left
MF_BARCODE_DIRECTION_BOTTOMTOP	Decodes it from bottom up

BarcodeInfo.Origin

Specifies the origin point of the decode area. This property is not used.

Type

Read/Write*:* *MfBarcodeOrigin*

BarcodeInfo.StartX

Specifies the starting point to decode in the x-coordinate. This property is not used.

Type

Read/Write*:* *int*

BarcodeInfo.StartY

Specifies the starting point to decode in the y-coordinate. This property is not used.

Type

Read/Write*:* *int*

BarcodeInfo.EndX

Specifies the ending point to decode in the x-coordinate. This property is not used.

Type

Read/Write*:* *int*

BarcodeInfo.EndY

Specifies the ending point to decode in the y-coordinate. This property is not used.

Type

Read/Write*:* *int*

Data

The decode result is set to MFBarcode.BarcodeData Class.
MFBarcode.BarcodeData Class is arranged as many as the number of decoded barcodes. If there is no decode result of barcode, NULL is set.

Type

Readonly*:* *MFBarcode.BarcodeData[]*

MFBarcode.BarcodeData Class

BarcodeData.Symbol

The symbol of read barcode is set.

Type

Readonly*:* *MfBarcodeSymbol*

Values

MfBarcodeSymbol	Barcode
MF_BARCODE_SYMBOL_CODABAR	Codabar
MF_BARCODE_SYMBOL_CODE128	Code128
MF_BARCODE_SYMBOL_CODE39	Code39
MF_BARCODE_SYMBOL_ITF	ITF
MF_BARCODE_SYMBOL_EAN_JAN	JAN13(EAN), JAN8(EAN)
MF_BARCODE_SYMBOL_UPC_A	UPC-A
MF_BARCODE_SYMBOL_UPC_E	UPC-E

BarcodeData.Direction

The decode direction is set.

Type

Readonly*:* *MfBarcodeDirection*

Values

MfBarcodeSymbol	Barcode
MF_BARCODE_DIRECTION_LEFTRIGHT	Decodes it from left to right
MF_BARCODE_DIRECTION_TOPBOTTOM	Decodes it from top down
MF_BARCODE_DIRECTION_RIGHTLEFT	Decodes it from right to left
MF_BARCODE_DIRECTION_BOTTOMTOP	Decodes it from bottom up

BarcodeInfo.StartX

The x-coordinate of started decode is set.

Type

Readonly*:* *int*

BarcodeInfo.StartY

The y-coordinate of started decode is set.

Type

Readonly: *int*

BarcodeInfo.EndX

The x-coordinate of ended decode is set.

Type

Readonly: *int*

BarcodeInfo.EndY

The y-coordinate of ended decode is set.

Type

Readonly: *int*

BarcodeData.Data

The barcode data that can be recognized is set.

Type

Readonly: *byte[]*